Situated Observation and Participation in Multiple-Agent Systems
by
Jefferson D. Montgomery
B.A.Sc., University of British Columbia, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in
THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia
July 2003
© Jefferson D. Montgomery, 2003
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

September 30, 2003
Abstract

A situated agent is not only embedded within its environmental system, but forms an integral and active component of the system as a whole. Accordingly, there are numerous requirements that a situated agent must meet including synchronization with and responsiveness to the system dynamics as well as appropriate and proactive operation — a non-trivial challenge made difficult primarily due to varying environment states and dynamics, difficult tasks with ambitious requirements, and noisy or ambiguous sensory and background information.

This thesis addresses these issues by introducing an agent architecture with an interruptible and modular design capable of producing quality-varying solutions based on constraints. The architecture adaptively schedules deliberation processes which enables the agent to evolve its action with the natural frequency of the environment's dynamics.

Resource-bounded deliberation is attained by attributing intentions to the agent, posed in the form of constraints, that influence its action. The thesis demonstrates how these constraints can be designed to achieve complex behaviours. Further, it shows that abstraction of behaviours based on a theory of intentionality provides a comfortable and tractable model not only for behaviour generation but also for behaviour recognition. The theory of intentionality is transcribed into a dynamic, probabilistic model that describes multiple, intentional agents and their environment. It is shown how the properties of intentionality, so transcribed, produce a computationally attractive model that allows real-time approximate inference.

Ultimately, a novel agent is introduced for the particular multiple-agent domain of robot soccer. The agent is implemented and tested to study the reactivity of the architecture, accuracy of the environmental modelling, and effectiveness of its deliberation.
Contents

Abstract ii

Contents iii

List of Figures vi

List of Tables ix

1 Introduction 1
  1.1 Reading guide 3

2 Background 5
  2.1 Agent architectures 6
    2.1.1 Constraint Nets 7
    2.1.2 The intentional stance 9
  2.2 Probabilistic modelling 11
    2.2.1 Inference 12
    2.2.2 Parameter estimation 14
  2.3 Internal representations: modelling the world 16
    2.3.1 Markov state traces 17
    2.3.2 Imperfect perception 18
    2.3.3 Self-localization 19
    2.3.4 Identifying objects 22
    2.3.5 Modelling the behaviours of other agents 22
  2.4 The RoboCup simulation league 26
    2.4.1 Object dynamics 27
    2.4.2 Stimulation 27
    2.4.3 Internal timing 29
    2.4.4 Action failure 30
    2.4.5 Interacting with the SoccerServer 30
List of Figures

2.1 Graphical representation of Constraint Net components .......... 8
2.2 Graphical representation of a Constraint Net module ........... 8
2.3 A Constraint Net model of a single-agent system ............... 9
2.4 A Markov chain model of vector evolution .................. 18
2.5 A hidden Markov model of dynamic perception ................. 19
2.6 Landmarks in the RoboCup Simulation League domain ......... 20
2.7 Localization using two landmarks ................................ 20
2.8 The "Two-landmark Uniform" Localization sensor ............... 21
2.9 Modelling agent action ...................................... 23
2.10 A mixture of hidden Markov models .......................... 24
2.11 A switched state-space model ................................. 24
2.12 The dynamic belief network representation of a hierarchical hidden Markov model ...................................... 25
2.13 The SoccerServer Constraint Net module .................... 27
2.14 Event timing within the SoccerServer module ................. 29
2.15 Typical fluctuations in the true and observed SoccerServer cycle .... 32
2.16 Comparing the Internal and External synchronization algorithms .... 33

3.1 A Constraint Net module of the constraint-based architecture ... 36

4.1 A hierarchical, probabilistic model of multiple, intentional agents acting in the world .............................. 42
4.2 Commitment and management of individual intentions, showing context-specific independence ......................... 45
4.3 Depiction of the model's secondary structure ................ 49
4.4 Independence relations among sets of modelled random variables .. 51
4.5 An approximate representation of a ring sector ................ 53
4.6 An example of Bayesian filtering ............................... 54

5.1 A Constraint Nets model of the RoboCup Simulation League system 60
7.1 A partially observable Markov decision process modelling a planning agent 102
## List of Tables

2.1 SoccerServer rules for effecting action. ........................................... 30

5.1 Inferring the validity of the optimal command time estimate assuming known cycle offset. ......................................................... 63

5.2 Inferring the status of pending action commands from the count of effected actions .................................................. 65

5.3 Rules used for labelling observations ............................................. 71

5.4 Description of agent actions ......................................................... 74

5.5 Summary of constraints .................................................................. 74

6.1 Comparison of errors obtained in the Static Self–Localization experiment ................................................................. 87

6.2 Localization errors obtained in the Dynamic Self–Localization experiment ................................................................. 88

6.3 Localization errors obtained in the Object Tracking experiment .... 89

6.4 The action confusion matrix produced by the hand-crafted model .. 96

6.5 The action confusion matrix produced by the best learned model ... 96
Chapter 1

Introduction

In a reception setting, a good waiter is one who always brings hors d'oeuvres to people who desire them, without bothering those distracted in conversation. Sometimes, however, distinguishing these states can be difficult (particularly as they are neither exclusive or covering). Therefore, a significant constituent of a waiter's job is analyzing customer's behaviour, as subtle as they may be. But waiters are not the only agents who benefit from an understanding of how their environment behaves. An understanding of behaviour can both explain and predict not only concrete interests such as the actions of fellow agents, but also characteristics like their preferences and commitments. For these reasons, behaviour analysis is an important precursor to any decision that relates to interaction with another entity.

However, there are significant hindrances to performing such detailed analysis in time-critical or ambiguous scenarios because causal abduction of behaviour constitutes a difficult problem that requires significant resources. This thesis considers situated observers: agents who perceive and act as non-trivial participants in their environment. A waiter is a situated observer — it is an agent situated in a reception environment with other agents, and it needs to both observe the other agents (interpreting their actions) as well as perform its own actions (to restock food, serve guests, and avoid other waiters among other tasks).

Developing such a detailed understanding of the scenario by interpreting the environment is a considerable task for a resource-bounded agent who must also devote sufficient resources to processes such as perception, reasoning, and action; which are fundamental to appropriate operation. The logistical difficulty leads to considering deliberation and modelling value: is it worth considering more options to make a better decision? At some point, devoting more resources to deliberation in order to better understand the situation may actually decrease the value of the resulting decision [RW39, BD94, DBS94, ZR96].

In sufficiently dynamic environments, the competing forces of decision value
and the cost of inaction counteract; a phenomenon that a successful agent must actively consider. The first part of this thesis introduces such an agent, designed around a notion of adaptive reactivity where each agent process is interruptible by a meta-level reasoner that schedules resources. This reasoner implements a novel scheduling algorithm that interprets percepts using an action failure model and allows the agent to infer whether it is being sufficiently reactive for the current environment dynamics — an inference that directly influences the action scheduling decision. When an action is desired, all the other processes combine to produce the best action discovered so far. In this way, the satisfaction of the reactivity constraints upon the situated agent is ensured.

Yet, the design of useful, interruptible processes which constitute the agent's architecture and combine to produce intelligent behaviour is non-trivial. This thesis will introduce a constraint-based architecture, wherein each constituent is posed in terms of constraints which are solved concurrently and in an interruptible manner; and demonstrates that three main processes (scene interpretation, means-ends reasoning, and planning) can be implemented with sufficient collective performance in a demanding task.

This architecture will be shown to provide a suitable framework for providing and managing resources to allow behaviour analysis of the external environment by a complex agent operating in a dynamic environment. However, performing behaviour analysis in itself is non-trivial — analyzing behaviour can be so complex that a punctilious description of general behaviour is infeasible. Even restricted domains such as hockey and soccer exhibit incredibly rich interactions.

One approximation commonly used by people is to project a complicated behaviour onto their own experiences and use their own behaviour as a description. For example, one may anthropomorphically say that a robot is lonely, bored, or hungry, or that it is collaborating with a comrade who has a common desire when actually quite different motivations are contributing to the observed behaviour [Bra84]. Researchers have proposed this method as a general behaviour description; even the actions of such devices as water valves and telephones can be described in terms of beliefs and intentions ascribed to them [McC79, Den87, CL90]. This approximation is believed to be more easily understood, leading to efficient design, modification and maintenance of behaviour models; but at some level these benefits are overcome by the greater accuracy of descriptions based on more realistic causal models. Ultimately, a projection to the observer's experiences can not be as accurate as a correct understanding of the behaviour's cause. Using Occam's razor, these approximate descriptions based on intentional ideas can only be considered legitimate if they provide the most succinct — yet still useful — description of the agent in
This thesis proposes that probabilistic models of multiple agents’ behaviour based on a projection to assumed intentional mental attributes can accurately interpret the environment using limited resources. The underlying hypothesis is that these mental attributes provide a useful, succinct description of the behaviour. To test this hypothesis, a reactive situated agent will be developed that uses a probabilistic intentional model of a multiple-agent environment to interpret its observations. The model is based on a theory of joint and individual intentions that considers the actions, beliefs, desires, and intentions of groups of agents, and includes rigorous modelling of the Robot Soccer World Cup (RoboCup) Simulation League (RSL) domain. This agent’s internal representations will be compared with the true state of the environment (known to the experimenter but hidden to the agent itself) as it engages in various tasks, allowing a characterization of the model’s accuracy and of the agent’s success in dynamic situations with varying resource requirements and uncertain percepts. However, the ultimate step (making full use of this internal representation within the observer’s decision process) is left for future work. Therefore, some aspects of the hypothesis are only confirmed qualitatively. Future work will include fully incorporating the intentional environment model into the observer’s decision process, and quantitatively analyzing its value based on the utility of resulting action.

1.1 Reading Guide

The thesis is organized into six more chapters. Chapter 2 provides a foundation for the remainder of the thesis. Background information related to the organization of an agent’s internal system is provided along with probabilistic methods of modelling an environment and their use as an agent’s internal representation, and the example domain of robot soccer.

The thesis continues with novel work. In Chapter 3, a constraint-based architecture is introduced conceptually. The architecture focuses on issues of deliberation, observation interpretation, and the management of resources so that appropriate and timely action can be achieved in demanding environments. Architecture specifics are then provided in the next two chapters.

In Chapter 4, a probabilistic model of a multiple-agent environment is developed and theoretically analyzed. The model describes the environment and a group of agents acting within it, and is to be used as the internal representation of an observer. The inherent complexity of the domain is handled by assuming that the observed agents are intentional entities, and that their behaviour therefore con-
forms to the restrictions of the theory of intentionality. However, the observer's own intentionality, and intentional interaction with the group, is not modelled.

The architecture and model are applied in a specific soccer agent in Chapter 5. This example is suitable for exploring fundamental issues of agency within multiple-agent systems, and provides context for a more meticulous treatment of both the architecture and intentional representations. Ultimately, in Chapter 6, the soccer agent is used to experimentally verify the properties of both the architecture and the intentional model.

Finally, Chapter 7 outlines conclusions that can be made from the theoretical and empirical analysis along with possible avenues of future inquiry.
Chapter 2

Background

This thesis' topic is of agents, environments, and their interaction. The exploration will be from the point of view of a particular, rational agent that exists in an environment with other agents. A rational agent is any entity that acts in its environment and whose action is appropriate (i.e., contributes to achieving the agent’s goals) given its knowledge and beliefs [PMG98, RN95]. This can be stated more formally using utility theory (a utility measure can be used to gauge the appropriateness of any action based on the desirability of particular situations compared to others) where an ideal agent is one that acts to maximize its expected utility measure based on its internal knowledge and its percept history [PMG98, RN95].

This constraint implicitly defines the agent's policy — a complete mapping from situations into actions — which can be used to describe the agent. However, complicated tasks in complicated environments necessitate complicated policies. For example, the complexity of the situation-action mapping implemented by the human brain is vast. Such policies cannot be described by simple functions or tables.

To cope with such complexity, agents are modelled based on a particular structure or architecture, which constrains the mapping. For example, a policy may be described using a collection of processes characterizing perception, modelling, deliberation, and action. Section 2.1 outlines a number of issues related to the specification of an agent architecture, and introduces (in Section 2.1.2) the notion of the Intentional Stance, whereby complex behaviours are achieved by resource-bounded agents using mental attitudes such as belief and desire, and through the use of intentions.

Because it forms a fundamental influencer of action, it is important that an agent correctly interpret its percept trace with respect to its internal knowledge. In general, processes characterized by uncertainty, like perception, require a representation that addresses that uncertainty. Probabilistic models, introduced in Section 2.2, provide a sound theoretical framework for representing uncertainty, and
Section 2.3 details how an agent can use probabilistic models to determine the most probable situation.

Although these aspects of agency will be discussed in general scope, domain and task specifics ultimately refine the agent's design and it is important — in fact, necessary in most cases — to capitalize on this knowledge. Throughout this thesis, reference is made to the RoboCup Cup Simulation League (RSL) domain: a multiple-agent domain with a wealth of interaction complexity that requires deliberative, situated agents [KAK+97]. Various elements of the RSL domain are important to the development of a RSL agent, and are discussed in Section 2.4.

2.1 Agent Architectures

An agent architecture is a framework for designing and constructing agents, adherence to which will produce an agent with specific, desired properties. For example, ideal agents exhibit generality, rationality, reactivity, benevolence, social ability, autonomy, and versatility, among other characteristics [WJ95]. However, many of these properties are incompatible and a correct treatment is accordingly difficult.

Initial attempts were based on the symbol-system hypothesis, and were characterized by an iterative sequence of sensing, modelling, planning, and acting, wherein the state of the environment is translated into discrete, logical descriptions in order to fit existing ideas about deliberation [Nil80]. This, however, has proved largely inadequate, particularly in environments that contain uncertain or unpredictable elements: researchers found a clean signal-symbol interface elusive.

In a dramatic rethinking of the notion of agency, focus shifted toward situatedness, wherein the agent's connection with its environment is of primary concern [Mac93, Mor84, Bro91]. A host of architectures were developed based on this reactive or behaviour-based paradigm, each abandoning the generality and expected power of formal systems for increased reactivity [Bro86, Ark98, AC87, KR90]. These architectures rejected internal representations and formal reasoning, preferring a tight coupling of sensation and action through distributed, asynchronous processes. Initially, such principles proved surprisingly adept, but over time it became clear that their success did not scale up to highly complex systems [HP91].

A series of hybrid architectures have been developed that borrow from both reactive and deliberative architectures trying to achieve a working compromise between reactivity and proactivity [GL87, Fer92, SM94]. The idea of asynchronous layers of operation is commonly adopted from the reactive paradigm, but the layers are typically allowed to be more general: lower layers tightly couple sensors to actuators with adaptable transfer functions and feedback control mechanisms that
give the agent reactive abilities in its most critical situations; higher layers add increasingly abstract behaviours, reasoning, and representations. Most importantly, computational abstraction is embraced using fully general communication between layers. Yet, the protocol of these interactions often tend to be ad hoc and cumbersome, and global system characteristics can be difficult to analyze. However, the development of a hybrid architectural language called Constraint Nets provides a unifying framework that can be used to design, model, and verify both local and global aspects of these hybrid dynamical systems [Zha94, ZM95, ZM99].

See [WJ95, KMB98, ZCLC00] for a more complete survey of agent architectures; the third reference presents a survey of architectures used by RoboCup agents.

2.1.1 Constraint Nets

Most dynamical systems are based on constraints, i.e., the system is asymptotically stable at the solution set of particular constraints. This motivates problem solving via a constraint-based approach, which involves specifying a collection of constraints — relations that specify a particular situation (e.g., a solution to the problem) — and allowing a constraint solver to asymptotically approach a solution through a dynamic process. This method is conceptually clear, and often engenders easily tasked systems.

This attraction is a motivating factor behind the Constraint Nets model [ZM95]. Constraint Nets (CN) presents a unified theory with a rigorous, formal semantics of dynamic systems based on topological and abstract algebra structures. Such semantics capture the most general structure of dynamic systems, including the flexibility to define concepts in both discrete and continuous abstract domains with both synchronous and asynchronous event structures.

Formally, a Constraint Net is a triple composed of a set of locations, a set of transductions, and a set of directed connections.

A **location** (represented graphically as a labelled circle as shown in Figure 2.1) stores an abstract value, which can be accessed or modified by a transduction through a connection. The value can be continuous or discrete and can represent a variable or an event.

A **transduction** (represented graphically as a labelled rectangle) represents a function from its input ports to a unitary output port (represented as the head and tail of directed arcs respectively).

Transductions operate asynchronously in general, but can be controlled by events through special event ports (represented graphically by small circles...
incident upon a transduction). For example, the transduction $\delta(i; e_1, e_2, \ldots)$ delays the input trace, $i$, until one of the connected events, $e_1, e_2, \ldots$, occurs.

A connection (represented graphically as a directed arc) connects a location with a transduction port.

![Diagram of Constraint Net](image)

Figure 2.1: A Constraint Net is represented graphically as a bipartite graph with locations depicted as circles, transductions as rectangles, and connections as directed arcs. This particular CN represents the equations $l_0 = f_n(l_i, p_t)$ and $p_t = \delta(l_o; ev)$, which aggregate to $l_o(n) = f_n(l_i(n), l_o(n - 1))$ where $l(n)$ represents the value of location $l$ after $n$ occurrences of event $ev$.

Much of the power of CN comes from the abstract treatment of algebraic structures. This allows locations and transductions to coexist in harmony, despite differences in domain or period, and provides a comfortable method of functional abstraction. A module in CN is the collection of a finite set of transductions, locations, and connections that collectively act as an abstract transduction. Graphically, as shown in Figure 2.2, modules are depicted as labelled, rounded rectangles. When depicting the internal structure of a module, the rectangle is drawn using dashed lines and port locations are drawn as double circles. Modules can be aggregated via union, coalescence, or hiding operations to provide interaction hierarchies that increase the expressive power and ease of the modelling task.

![Diagram of Module](image)

Figure 2.2: Graphical representation of a Constraint Net module depicting the internal structure of module $Md$ (left) and the same module used in a CN (right). The feedback is considered implicit and internal to the $Md$ module.
Experiments provide evidence that the constraint-based CN approach is a clean and practical design framework for perceptual agents [BKL+93, Zha98, ZM99, Mon02]. The entire agent system (the agent’s controller, body, and the environment) can be modelled in CN [Zha94, PMG98] as in Figure 2.3 which models a single-agents system.

![Figure 2.3: A Constraint Net model of a single-agent system, depicting the agent’s controller and body along with its environment as separate modules.](image)

2.1.2 The Intentional Stance

Many interesting agents are far too complex to be described by simple sets of formal rules. When physical explanations are impractical, more powerful abstractions are needed to explain behaviour. This idea has resulted in shifts in popular psychological paradigms, rejecting stimulus-response rules of behaviour for more abstract notions such as functionalism. Though not logically required (and perhaps not even valid) such abstractions are legitimate in the sense that they can provide succinct, explanatory models of an agent’s behaviour, even when its policy is unknown [McC79].

The intentional stance is such an abstraction whereby behaviour is modelled through mental attitudes (such as believing, desiring, anticipating, etc) which are attributed to the agent [Den87]. However, the attribution is more than just anthropomorphism: mental attitudes and their interactions are strictly defined and the intentional stance abstraction also presupposes that agents will act in conformance with these defined influences.

Although the attribution of attitudes provides a convenient and familiar framework for an observer to describe, predict, and specify behaviour, the intentional stance model fails if the observed agent’s behaviour is generated by an inconsistent...
mechanism. There are, however, convincing reasons that a resource-bounded agent would conform to these rules. Agent beliefs, for example, are useful in modelling rational behaviour that may be otherwise counter-intuitive given the true state of the world. An agent's beliefs are propositions that capture its perspective on the world (which may be incorrect) and represents information immediately available to the agent [RG91]. Beliefs are distinguished from percepts through conscious control — seeing is not believing in this sense: beliefs and percepts can be inconsistent (and beliefs can be internally consistent as well). For example, I can see water falling from the sky, and still believe that it is not raining. Also, although I believe that it is not raining, I can simultaneously believe that I will get wet outside from the rain.

Also, because means-ends reasoning competes with reactivity in resource-bounded agents, these agents must constrain their deliberation in some way. Intentions perform this critical role in a general way by significantly reducing the domain of the agent's policy [Bra87, CL90, BIP98]. An intention is a commitment to bring about a particular proposition [Bra87, CL90]. The commitment is strong in that it persists until the agent believes it has succeeded, or never will (in which case the agent may abandon the intention). Further, this commitment encompasses the following implications upon further deliberation and means-ends reasoning:

1. Agents are bound to satisfy their intentions, i.e. any rational action must be performed in a direct attempt to achieve one of the agent's intentions, based on the current beliefs of the actor.

2. Candidate propositions cannot be adopted unless they are consistent with the current intentions (but agents need not intend the side effects of their intentions).

3. In some circumstances, agents believe that they will satisfy their intentions but they always believe that it is possible to satisfy their intentions and they never believe that they will not satisfy their intentions.

In multi-agent systems, agents interact through cooperation, coordination, or competition. These interactions can be described in a number of ways, but requires some form of mutual support and commitment between agents [Bra92, Sea90]. One such description is through the use of joint intentions [LCN90]. A joint intention is a commitment by a group of agents to bring about a proposition, which persists until the group comes to mutually believe in their success or ultimate failure (i.e., they will never succeed). The commitment constrains future deliberation and means-ends reasoning concerning group activity by requiring all joint intentions to be self-consistent as well as being consistent with each member's individual intentions.
Joint intentions are a natural extension of the intentional stance for single agent systems but the amount of interaction required to determine theoretical mutual belief and knowledge can be impractical. Implicit mutual belief can be assumed for groups of agents that are likely to share similar desires (among teammates for example) although the same cannot be said about dissimilar agents [BBS+00].

The intentional stance also has been used as a means of specifying behaviours by a number of successful agent architectures [BIP98, GL87, RG91]. However, this thesis will focus on its use as an explanatory model of agent behaviour.

2.2 Probabilistic Modelling

Realistic agents are subject to ambiguous and fallible sensors. For example, any two-dimensional projection of a three-dimensional environment is subject to various perspective ambiguities. Consequently, agents are often faced with decisions under uncertainty, and are only considered rational (in terms of decision theory) if they consider all of their options using probabilistic models and probability theory.

A probabilistic model is a collection of random variables that describe a situation. A random variable (noted as a bold, capital symbol such as $Z$) is a mapping from a specific outcome $z$ into a scalar measure and the probability of that outcome is noted as $Pr(Z = z)$.

For any collection of random variables, $Z_{1:N} \equiv \{Z_1, Z_2, \ldots, Z_N\}$, statistical correlations and dependencies can be fully described by a belief network$^2$ [Pea88, CDLS99]. A belief network (BN) is a directed, acyclic graph (whose nodes represent random variables and whose arcs represent conditional dependence) with prior and conditional probability distributions over the constituent random variables. A BN encodes a particular factorization of the variables' joint distribution. Specifically, the semantics of a BN, $B_0$, are defined by Equation 2.1 where Parent($Z_i; B_0$) is the set of variables who are parents of node $Z$ in graph $B_0$.

$$Pr(Z_{1:N}) = \prod_{i=1}^{N} Pr(Z_i | Parent(Z_i; B_0)) \quad (2.1)$$

When variables can change over time, time and its relation to the variables must be modelled either explicitly (by representing time as a specific random variable) or implicitly (by modelling transitions in the belief state over time). Dy-$^1$The notation $X_{a:b}$ and $X_a:b$ refer to the sets $\{X_a, X_{a+1}, \ldots, X_b\}$ and $\{X_a, X_{a+1}, \ldots, X_b\}$ respectively. If $a > b$, the set is empty and if $a = b$ the set contains only $X_a$ (or $X_a$).

$^2$Also called a Bayesian network or a probabilistic network.
namic belief networks provide a framework for the latter, focusing on the exploitation of temporal independence [Mur02]. A **dynamic belief network** (DBN) is defined as two BNs which aggregate to model semi-infinite sequences of random variables, \( Z_{0:1}^{1:N}, \ldots, Z_{t}^{1:N}, \ldots \): \( B_0 \) describes a prior probability function over the variables, \( \Pr(Z_{0:1}^{1:N}) \), and \( B_\rightarrow \) describes the variable's stochastic evolution over time, \( \Pr(Z_{t+1}^{1:N} | Z_{0:t}^{1:N}) \). These two networks define the full joint distribution of the variable sequence, based on the semantics of each individual BN, as Equation 2.3.

\[
\Pr(Z_{0:t}^{1:N}) = \Pr(Z_{0}^{1:N}) \prod_{\tau=0}^{t} \Pr(Z_{\tau}^{1:N} | Z_{0:\tau-1}^{1:N})
\]

(2.2)

\[
= \prod_{i=1}^{N} \Pr(Z_{0}^{1:N} | \text{Parent}(Z_{0}; B_0)) \prod_{\tau=0}^{t} \prod_{j=1}^{N} \Pr(Z_{\tau}^{1:N} | \text{Parent}(Z_{\tau}; B_\rightarrow))
\]

(2.3)

### 2.2.1 Inference

In general, the collection of modelled variables can be separated into two sets: an evidential set \( Y \) containing those variables who can be observed directly, and a hidden set \( Q \) containing those who have been hypothesized, \( Z = \{Y, Q\} \). General inference on a dynamic belief network is the determination of the joint distribution over unobserved variables conditioned upon specific evidence, \( \Pr(Q_0:1, Y_{t+1:1:oo} | Y_{0:t} = \tilde{y}_{0:t}) \). Motivated by the structure evident in Equation 2.2, useful inference in a DBN can be implemented iteratively by incrementally predicting future states and then updating the predictions from incoming evidence [RN95, Mur02]. This procedure, called *Bayesian filtering*, maintains the filtering distribution, \( \Pr(Q_t | Y_{0:t}) \), using the following two steps.

1. During the prediction step, a distribution over the random values at the next time step is calculated. At the initial time, this is the prior distribution which is specified as part of the model. At any other time, the distribution is calculated by considering possible transitions from the current filtering distribution:

\[
\Pr(Q_{t+1} | Y_{0:t}) = \int \! \! \int \! \! \int \Pr(Q_{t+1}, \text{Parent}(Q_{t+1}; B_\rightarrow) | Y_{0:t})
\]

(2.4)

2. To obtain the current filtering distribution, the prediction is then updated by incorporating current evidence into the current instantiation of \( B_0 \):
\[
\Pr(Q_t|Y_{0:t}) = \Pr(Q_t|Y_{0:t-1}) \frac{\Pr(Y_t|Q_t, Y_{0:t-1})}{\Pr(Y_t|Y_{0:t-1})}
\]  

(2.5)

A host of algorithms exist that perform inference analytically [Pea88, Rab89, Sha86, Sha90, CDLS99]. However, this is generally NP-hard in the size of the BN graphs [Coo90, RN95]. This means that for certain networks of sufficient dimension (large number of random variables or continuous random variables) or insufficient conditionalization (grid-like graphs or graphs containing nodes with high fan-in) exact inference takes prohibitive amounts of time. For situated agents, this is a real concern and approximate methods become necessary. These methods approximate distributions in such a way as to simplify the inference task — although inference remains NP-hard in the accuracy of the estimates [DGH92].

2.2.1.1 Sequential Monte Carlo Sampling

Sequential Monte Carlo sampling\(^3\) is a method for stochastically approximating an evolving probability distribution. The approximation is to represent the distribution with a collection of randomly located delta functions, fit to the distribution of interest [Dou98, DFG01, GRS96, FTBD01]. For example, the filtered distribution in a DBN’s current slice can be approximated using Equation 2.6:

\[
\Pr(Q_t = q|Y_{0:t}) \approx \tilde{Pr}(Q_t = \bar{q}|Y_{0:t}, N) \overset{\text{def}}{=} \sum_{i=1}^{N} w_t^{(i)} \delta_k(q - \tilde{q}_t^{(i)})
\]  

(2.6)

where \(\delta_k\) is the delta function, defined in Equation 2.7.

\[
\int_{\text{domain}(x)} \delta_k(x) \, dx = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{else} \end{cases}
\]  

(2.7)

The approximation is parameterized by the path of \(N\) pairs, or particles: \(\bar{q}_t^{(i)}\) (which are sampled randomly from a proposal distribution, \(p(Q_{0:t}|Y_{0:t})\), whose support is a superset of the support of \(\Pr(Q_{0:t}|Y_{0:t})\)) and \(w_t^{(i)}\) (which weight the importance of each sample according to the true distribution via Equation 2.8).

\[
w_t^{(i)} \propto \frac{\Pr(Q_{0:t} = \bar{q}_t^{(i)}|Y_{0:t} = \bar{y}_{0:t})}{p(Q_{0:t} = q_t^{(i)}|Y_{0:t} = \bar{y}_{0:t})}
\]  

(2.8)

\(^3\)Also known as sequential importance sampling, particle filtering, or the CONDENSATION algorithm.
Ideally, one would sample from the filtered distribution itself, but difficulties in sampling from the distribution of interest lead to different proposal distributions. Because the system dynamics model is often easily parameterized, it is commonly used as the proposal distribution: \( p(Q_{0:t}|Y_{0:t}) = Pr(Q_t|Q_{t-1}, Y_{0:t}) \) [IB98]. However, when observations are highly informative (more so than the dynamics) it is beneficial to sample from the posterior — \( p(Q_{0:t}|Y_{0:t}) = Pr(Q_t|Q_{t-1}, Y_{0:t}) \) — which minimizes importance weight variance and tends to focus the sample locations onto more informative regions of the distribution of interest [FTBD00, Dou98].

Monte Carlo sampling approximations are important because, as opposed to using a different but similar-looking model, the nature of the approximation is to focus consideration on a subset of the original model, which is represented with varying resolution and is refined over time. It can be proved that the limit of the approximation goes to the true distribution (Equation 2.9). Although the maximum likelihood and maximum a-posteriori values of the model remain elusive, the easily calculated mean of the approximated distribution is the only statistic of interest in this thesis.

\[
\lim_{N \to \infty} \Pr(Q_{0:t}|Q_{0:t}; N) = \Pr(Q_{0:t}|Y_{0:t})
\]  

(2.9)

2.2.1.2 Rao-Blackwellization

For some models, the opportunity exists to develop a hybrid inference technique, wherein a subset of the filtering distribution can be analytically marginalized, leaving a lower-dimensional approximation [AK77, DdFMR00]. This so-called Rao-Blackwellized estimate is theoretically not less accurate than a strictly sequential Monte Carlo estimate with the same number of samples [Dou98] but has shown empirically to significantly reduce the variance from the true filtering distribution in most cases.

2.2.2 Parameter Estimation: the Expectation Maximization Algorithm (EM)

Learning a probabilistic model involves estimating the structure (i.e., the conditional dependencies) and the parameters (i.e., the prior and conditional probability functions) of a set of random variables. The discussion in this thesis is restricted to the case of known structure wherein some of the random variables of interest cannot be observed in the system. In this case, the problem is to estimate the parameters, \( \Theta \), of a model with known structure from a set of data, \( D \), that is supposedly generated by it.
However, exact Bayesian learning is often intractable. Useful, sub-optimal approximations can be derived using techniques such as gradient descent optimization of the posterior or likelihood surface\textsuperscript{4} [BKRK97, Thi95]. One such approach is the Expectation Maximization algorithm (EM) which is proved to converge and has both online and offline formulations [DLR77, NH98, Lau95].

Assuming that the observed history of the system is comprised of independent and identically distributed sequences \((D = \{y_{1:T_1}, y_{1:T_2}, \ldots, y_{1:T_M}\})\) then the likelihood is the product of likelihoods from each sequence. Since the maximum of the likelihood’s logarithm and the maximum of the likelihood itself are co-located at the same parameter values, the log-likelihood may be considered — replacing the product with a sum over the log-likelihoods of each data sequence.

The key to the EM algorithm is to iteratively adapt the parameter estimate along with a distribution, \(Q\), in such a way that a lower bound on the log-likelihood monotonically increases toward a local maximum. For example, the negative free energy, \(F\), of the system is a lower bound based on the concavity of the logarithm relation.

\[
\log \Pr(D|\Theta = \theta) = \sum_{i=1}^{M} \log \int \cdots \int \Pr(Q_{0:T_i}, Y_{0:T_i} = y_{0:T_i}; \theta) \geq \sum_{i=1}^{M} \int \cdots \int Q(Q_{0:T_i}) \left[\log \Pr(Q_{0:T_i}, Y_{0:T_i} = y_{0:T_i}; \theta) - \log Q(Q_{0:T_i})\right] = \sum_{i=1}^{M} F_i(Q; \theta)
\]

This lower bound \((\log \Pr(D|\Theta) \geq \sum_{i=1}^{M} F_i(Q; \Theta))\) can be iteratively improved using the following procedure, which guarantees that the bound will never decrease.

Initialize \(Q_1\) and \(\theta_1\).

Repeat (increasing \(k\) and cycling \(i\) through \([1, M]\)),

\textsuperscript{4}If a priori knowledge is uninformative, the maximum likelihood estimate approaches the maximum a-posteriori estimate in the limit of infinite amounts of unique training data [Gha98] (a direct result of Bayes' theorem).
\[ Q_{k+1} = \arg\max_Q \mathcal{F}_i(Q; \theta_k) \quad (2.13) \]

\[ \theta_{k+1} = \arg\max_{\theta} \mathcal{F}_i(Q_{k+1}; \theta) \quad (2.14) \]

Until convergence.

It can be shown that the distribution that minimizes the free energy for fixed parameters (the ‘E’ step of the algorithm) is the posterior over hidden variables\(^5\) [Bil97]. Therefore, the iteration can be conglomerated into one step — Equation 2.15 — which can be described as iterative maximization over the expected values of local terms involving each node and its parents with respect to the posterior distribution over hidden variables:

\[
Q_{k+1}(Q_{0:T_i}) = \Pr(Q_{0:T_i}|Y_{0:T_i} = \bar{y}_{0:T_i}; \theta_k) \quad (2.15)
\]

\[
\theta_{k+1} = \arg\max_{\theta} \int \ldots \int \Pr(Q_{0:T_i}|Y_{0:T_i} = \bar{y}_{0:T_i}; \theta_k) \log \Pr(Q_{0:T_i}, Y_{0:T_i} = \bar{y}_{0:T_i}; \theta)
\]

In general, these expectations are intractable to compute. However, a general, approximate solution can be obtained by randomly sampling a finite set of hypotheses from \( \Pr(Q_{0:T_i}|Y_{0:T_i}) \) [WT90]. By only considering those hypotheses, the integral reduces to a finite sum, significantly reducing complexity. This Monte Carlo integration converges to the true solution in the limit of infinite samples but monotone convergence can otherwise not be guaranteed [Bis94]. In the unruly case where the posterior distribution over hidden variables cannot be sampled tractably, a more complicated sampling technique such as Markov Chain Monte Carlo or Gibbs sampling [GRS96] must be used.

### 2.3 Internal Representations: Modelling the World

The solution of most difficult problems benefits from reasoning about the problem’s domain at some level of abstraction. For example, there is intuitive information that a soccer-playing agent would require to solve the problem of getting the ball into a

---

\(^5\)Variational methods use other intuitive distributions and try to minimize the distance from this posterior. Such maximization is approximate, and is only used when expectations over the posterior can not be computed, nor sampled, in a tractable manner.
goal: the locations of the ball and the goal, and how to move the ball. Two types of information can be distinguished: procedural knowledge concerning action (for example, how to move the ball) and environmental knowledge concerning the state of objects in the environment. Both types of information are relevant to the agent’s control, but only environmental knowledge need be included in its representation of the domain (procedural knowledge is used to design the agent’s controller).

The world state is a vector of all the information deemed relevant to the agent’s control. In the RoboCup Simulation League, for example, the world state is the 136-dimensional vector specified below:

\[
\begin{align*}
\vec{x}^a &\in \text{field} \subset \mathbb{R}^2 \\
\vec{x}_i^p &\in \text{field} \subset \mathbb{R}^2 \\
\vec{x}^b &\in \text{field} \subset \mathbb{R}^2 \\
\vec{v}^a &\in \mathbb{R}^2 : \|\vec{v}^a\| \leq s_{\text{max}}^a \\
\vec{v}_i^p &\in \mathbb{R}^2 : \|\vec{v}_i^p\| \leq s_{\text{max}}^p \\
\vec{v}^b &\in \mathbb{R}^2 : \|\vec{v}^b\| \leq s_{\text{max}}^b \\
\theta^a &\in [0, 2\pi) \subset \mathbb{R} \\
\theta_i^p &\in [0, 2\pi) \subset \mathbb{R} \\
\phi^a &\in [-\phi_{\text{max}}, \phi_{\text{max}}] \subset \mathbb{R} \\
\phi_i^p &\in [-\phi_{\text{max}}, \phi_{\text{max}}] \subset \mathbb{R} \\
\end{align*}
\]

- The agent’s position.
- Player i’s position (21 players).
- The ball’s position.
- The agent’s velocity.
- Player i’s velocity.
- The ball’s velocity.
- The agent’s heading.
- Player i’s heading.
- The agent’s head direction.
- Player i’s head direction.

It is up to the agent to then determine the content of the world state during operation. However, it may not be possible for the agent to do this unequivocally as sensor readings can be ambiguous or erroneous, and purely causal models can be inaccurate. In general, an agent may consider a probability distribution over possible world states. This is the agent’s own belief about its external environment, and is governed dynamically by what the agent is able to infer through observation (interpreting sensations using sensor knowledge) and prediction (using causal knowledge to judge future states). In real environments, neither is sufficient in and of itself because the agent’s knowledge is inaccurate and sensor data is unreliable, but performance is greatly improved by combining both of these sources.

2.3.1 Markov State Traces

If the state of the world changes over time, there is a trace of agent beliefs corresponding to the world’s evolution. Equipped with knowledge of its causal evolution, an agent can use a mechanism of prediction to extend and correlate hypotheses over time. To make this prediction computationally feasible, the agent can assume that the trace possesses the Markov property: the evolution of a vector
is Markovian (in the first order) if its value is independent of its history, given its immediate predecessor. This can be written as $X_{t+1} \perp X_{0:t-1} \mid X_t$ — which means $Pr(X_{t+1} \mid X_{0:t}) = Pr(X_{t+1} \mid X_t)$ — or graphically as in the DBN of Figure 2.4.

An agent can hypothesize the future state of a Markovian model by iteratively advancing a prior distribution through a relatively simple stochastic evolution model: the locality imposed by the Markov property can significantly reduce prediction complexity. However, it is always important to consider whether a model is consistent with the environment’s true dynamics. For example, an object’s position alone is not likely to exhibit Markovian evolution; whereas in conjunction with its velocity, applied force, coefficient of friction, etc., the Markov property is more plausible. Any domain can be modelled as Markovian by using an appropriate world state (ultimately, the entire history can be included in the state description).

2.3.2 Imperfect Perception

In general, agents are not equipped to fully observe their environment. When some part of the state is ambiguous or hidden from the agent, it must be inferred through prediction (as described in Section 2.3.1) as well as perception. Perception is the process of mapping the world state, evident in various stimuli, into percepts. This mapping depends on the type and quality of the agent’s sensors but is generally neither one-to-one nor deterministic: percepts represent equivocal information about the state of the world. In general, perception is a stochastic process described by a perception model that is used by an agent to describe the relation between percepts and the world state and to make prioritized judgements regarding observations.

As a part of the perception model, it is common to assume that percepts depend only on the current world state ($Y_t \perp X_{0:t-1}, Y_{0:t-1} \mid X_t$). This assumption is often accurate and can significantly reduce the complexity of interpreting percepts. Combining the Markov property with this assumption, produces the familiar hidden Markov model, shown in Figure 2.5.

Despite these assumptions, however, realistic models are complex and per-
forming exact inference can still be intractable. This problem has motivated the use of approximate models (obtained by fitting the model to a parametric distribution of lower order) that allow computable inference. Common examples are Gaussian distributions [Kal60], delta functions (as described in Section 2.2.1.1), and neural networks.

2.3.3 Self-Localization

The theory in Section 2.3.2 describes how to filter multiple observations over time where, as the state changes, the percept trace may contain new information that increases the belief accuracy. A similar effect is produced when multiple sensors observe the same object at the same time; because different sensing modalities may contain different information, combining these observations often decreases the ambiguity of any individual sensor.

A special case of observation using multiple percepts is self-localization using landmarks. Landmarks are objects whose position is assumed to be known by the agent. For this assumption to hold, these objects are usually chosen to be stationary, easily identifiable objects (for example, the flags (disks), goals, and field lines shown in Figure 2.6).

Observing landmarks with agent-centred sensors is equivalent to observing the transformation between the agent and the world, i.e., the agent's position and orientation or pose. For example, one range observation constrains the agent to lie on a circle centred on the landmark; the range to two different landmarks constrains the agent to lie on the intersection of two such circles (one of two points); etc., as shown in Figure 2.7.

Estimates of this type have been commonly used for localization in the RoboCup Simulation League (RSL) where each range measurement, \( i \), is aggre-
Figure 2.6: Landmarks locations in the RoboCup Simulation League domain. Figure adapted from [CFH+02].

Figure 2.7: The range to two landmarks constrains the agent to lie on one of two locations, which can be disambiguated using other landmarks. Range measurements with bounded noise constrain the agent to the shaded region.
gated with noise of a known bound $d_i$. It is also common to assume that this noise is uniform over the area of intersection. This area can take on complicated shapes (depending on the locations of the two landmarks and the agent) so it is typical to fit the area to a two-dimensional, symmetric Gaussian with variance $\frac{1}{12}(d_1^2 + d_2^2)$, and then to use this variance to weight the estimate [dBK02].

The agent could benefit from a more complicated model of perception. Although there has been minimal application of Bayesian filtering to the RSL domain, [dBK02] implements a Bayesian filter for agent position (but neglects filtering any other dimension of the state). They model a localization sensor using two cues:

1. the weighted average of every possible two-landmark estimate as described above, and

2. bounds on possible locations (obtained by intersecting the rings of possible agent locations from each landmark range observation).

Their localization sensor is then a uniform distribution whose domain is a square centred on the two-landmark estimate intersected with the bounds on possible locations, as illustrated in Figure 2.8. Although this uniform sensor model yields a tractable, exact Bayesian filter, they use an approximate filter based on a sampled distribution. Although this is more computationally expensive and less accurate than the exact filter, they report highly accurate beliefs in pose [dBK02].

Figure 2.8: Localization sensor described by [dBK02]: the “Two-landmark Uniform” localization sensor models the agent location as being uniformly distributed within the shaded area (the intersection between a square centred on the two-landmark estimate and the bounds on possible locations).
2.3.4 Identifying Objects

When an agent observes multiple, independent objects, it is presented with a data association problem: which observation corresponds to which element of the state. In many cases, sensors are not able to easily identify the object they perceive, yet it has been observed that a lot of these ambiguities can be resolved over time by considering causality [Int99, BSF88, RH98]. Specifically, an agent can correlate the modelled equations of dynamics with observations to maintain multiple hypotheses over time and eliminate hypotheses that become sufficiently improbable. Classical multiple hypothesis tracking represents multi-modal distributions using a set of Kalman filters, each corresponding to an association between a mode and an observation [BSF88, CH96]. Alternatively, one could use a sample-based representation or some hybrid. In practise, an agent is usually able to eliminate competing hypotheses quickly.

2.3.5 Modelling the Behaviours of Other Agents

Plan recognition is the inverse of planning — not concerned with determining optimal behaviour, but of inferring an agent's conduct influencers after observing its action. However, these inverse problems often share a common model of behaviour: a probabilistic policy\(^6\). A policy is a general model of action selection — mapping state-action pairs into a desirability measure used to select action. This has a direct Bayesian interpretation that can be used to elaborate the evolution of the world state, as is done in Figure 2.9, providing a useful framework for modelling behaviours: plan recognition can be posed as probabilistic inference in this type of model. This section will present several such models with varying generality.

The hidden Markov model (HMM), shown in Figure 2.5, is a general dynamic extension of mixture models that can model any distribution with arbitrary dynamics [Rab89]. Therefore, there is nothing theoretically problematic with using HMMs for modelling general behaviour, for example by expanding the world state description to include some characterization of action. However, the time and sample complexity of HMMs are exponential in the dimension of the state space — the so-called curse of dimensionality. This makes inference impractical, in general, and subjects learning to severe over-fitting problems as training data is unlikely to represent the full structure representable by the model. These issues are critical when

---

\(^6\) Alternatively, one could recognize behaviours using deterministic, formal descriptions of plans [Gre69, MH69, LRL+97]. The power of these models, however, do not translate well to situated systems which require physical grounding. The case for probabilistic inference in plan recognition is further argued in [CG93].
modelling multiple classes of action sequences, which can dramatically increase the state-space despite their inherent structure.

Indeed, a general supposition of plan recognition is that behaviour is modular and reuse of particular action sequences are inherent properties of most agent domains: complex sequences can be described using a sequence of piecewise simple sequences with simple interdependencies. Nonetheless, intuitive constraints based on such hierarchical or modular structure of the system cannot be easily exploited by the HMM. Consequently, HMMs alone are only suited for modelling simple, individual behaviours [HV00].

For more complicated cases, solutions can sometimes be obtained by discretizing the space into piecewise-tractable sections governed by intermediate-level abstract descriptions such as goals, plans, and abstract policies. The most obvious discretization may be to model this type of behaviour using a mixture of HMMs where the mixture component is allowed to transition upon completion of an individual HMM [Smy97, HoeOl, NH99, KA94] (as shown in Figure 2.10) or to transition with the state vector (a switched state-space model [PRM01, PFH99, PRCM99], as shown in Figure 2.11).

An alternative is the hierarchical hidden Markov model (HHMM) which is specifically designed for domains that exhibit hierarchical structure with dependencies at multiple scales [FST98]. The HHMM is a stochastic automaton with three distinct types of states.

**Production states** emit a single observation.

**Abstract states** emit a sequence of observations governed by a sub-automaton.

No horizontal transition may be made without first entering a sub-automaton.

**Termination states** return focus to the parent, abstract state. Because the termination of each sequence class is modelled explicitly, the model can be trained
Figure 2.10: A mixture of hidden Markov models handles dependencies at multiple scales by explicitly modelling the class type responsible for each sub-sequence. Class transition occurs after the termination of the sub-HMM, which is modelled using a determined sequence length. Inference based on hard-coded sequence termination is easy (although it is also sensitive to noise in sequence length) but learning can only be done on pre-segmented sequences. Shown are two sequential classes with three time steps each.

Figure 2.11: Three time steps of a switched state-space model. A switched state-space model alleviates the strict, fixed-width requirement of the mixture of HMMs model, allowing a learning algorithm to classify the space based on observed evidence. Each subsequent classifier emits a single observation, as opposed to a fixed-length vector, because the class variable is allowed to transition at every time-step. However, there is no specific mechanism for forcing the top level to change more slowly than the bottom level, or to change based on other cues such as the satisfaction of particular goals.
As sub-automatons are called recursively, the calling context is maintained in a depth-limited stack. This method of handling recursion is computationally efficient but bears a less powerful model than those which allow infinite recursion such as stochastic context-free grammars. However, a significant advantage of the HHMM is that it can be represented as a DBN thereby reducing inference complexity from cubic to linear in the number of states [MP01]. The resulting generic HHMM, represented as a DBN, is shown in Figure 2.12.

![Dynamic Belief Network](image)

**Figure 2.12**: Three slices of a dynamic belief network representing a hierarchical hidden Markov model. The model is very much like the switched state-space model (Figure 2.11) save the random variables $F^i$ which model transitions within the sequence hierarchy.

The abstract hidden Markov model (AHMM) is an adaption of the HHMM tailored to the domain of single-agent policy recognition [BVW00, BVW01]. The space is discretized by modelling multi-scale behaviours with *abstract policies* [SPS99]. Abstract policies map the state of the world into other abstract policies, instantiating more and more refined policies until a final concrete action is effected.

Computational efficiency is achieved by assuming that the state of an abstract policy depends only on the state of its immediate parent policy, and its ter-
mination probability depends only on the current policy, not on the whole context. This substantially reduces the number of parameters and allows for more efficient approximate inference. However, this model has not been tested in multiple agent environments and there is no clear notion (either in the model or in the theory of abstract policies) governing the interactions of multiple agents. Simply replicating the model is likely too general a solution to yield tractable or useful results.

2.4 The RoboCup Simulation League

In the study of situated agents, a consideration of their environment and task domain is of equivalent value to their architecture and internal processes. This thesis concentrates on agents situated in the RoboCup Simulation League (RSL) domain.

[Mac93] proposed robot soccer as a suitable domain for studying situated agents, since the domain required them to overcome the restrictive assumptions of so-called "Good Old Fashioned Artificial Intelligence and Robotics." This idea was later posed as a formal challenge to artificial intelligence researchers and — with the advent of a yearly showcase of their successes (the Robot World Cup [KAK+97]) — has become a paradigmatic domain for analyzing and constructing situated, multiple-agent systems.

Robot soccer is suitable because soccer agents are situated in a complex and highly dynamic environment. Successful operation requires careful consideration of the agent's percept trace and development of complicated individual policies and multi-agent strategies while acting quickly enough to properly interact with the ball and other players. Because these factors are so crucial in this domain, the properties of the control algorithms can be clearly observed and compared.

The RSL system is composed of multiple, distinct software programs that interact through a lossy communication channel: the Agent programs that implement the agent controllers for 22 agents, and the SoccerServer program that implements the simulated environment including the ball and referee. The SoccerServer module (which can be modelled in Constraint Nets as in Figure 2.13) simulates the effects of action on its world state and provides stimuli to each agent [Nod95, CFH+02]. It is critical for each agent that is interacting with the SoccerServer to have sufficient knowledge about its operation. In particular, knowing

1. how action is effected in the simulated environment, as well as
2. the relation of the stimuli to the environment state

allow the agent to best reconstruct the world state. This knowledge can be derived from the SoccerServer's internal timing, rules for effecting action, stimulation
mechanism, and equations of dynamics.

Figure 2.13: The Constraint Net module modelling the SoccerServer.

### 2.4.1 Object dynamics

The Simulate module determines the next state of the ball, each agent, and the game. The state of the game (based on the rules of soccer) is reported to each agent through a hear percept. All other objects in the world adhere to the same parametric motion model described by the following equations of discrete dynamics, where at time step $t$, $\vec{x}_t$ and $\vec{v}_t$ are the position and velocity of the object respectively, $\vec{f}_t$ is some external force applied instantaneously to the object, and $\mu$ is an object-specific parameter describing collective friction [Nod95, CFH+02].

$$\vec{x}_{t+1} - \vec{x}_t = \vec{v}_t \quad (2.17)$$

$$\vec{v}_{t+1} - \vec{v}_t = -\mu \vec{v}_t + \frac{1}{m} \vec{f}_{t+1} \quad (2.18)$$

### 2.4.2 Stimulation

The SoccerServer provides both proprioceptive and visual stimuli to each agent. Proprioceptive stimulus is provided through the Stimulate body module, and contains

- the agent’s current stamina,
- the agent’s current effort level,
- a count of effected actions,
a measurement of the agent's current head direction relative to its body, $y^a$, based on its actual value, $\phi^a$, in radians\(^7\) according to Equation 2.19, and

$$y^a = [\phi^a - \frac{\pi}{360}] \quad (2.19)$$

- a measurement of the agent's previous velocity, $\bar{y}^a$, based on its actual value, $\bar{v}^a$, according to Equation 2.20.

$$\bar{y}^a = \left[ \frac{100(1 - \mu)\|\bar{v}^a\| - \frac{1}{2}}{100(1 - \mu)} \right] \cos \left( \frac{\pi}{360} \right) \quad (2.20)$$

Where

$$a \leq b \overset{\text{def}}{=} a \left[ \begin{array}{c} \cos \theta \\ \sin \theta \end{array} \right]$$

Visual stimulus is provided through the Stimulate vision module by a typical\(^8\) sensor that measures an object's position relative to the observer [Nod95, CFH+02]. The precise mapping from world state to percept for the position any object, $o$, is defined by Equation 2.21, where $q$ is a SoccerServer parameter whose default is 100 for landmarks and 10 for all other objects.

$$\bar{y}^o = \frac{1}{10} \left[ 10 \exp \left\{ q \left[ \ln \|\bar{x}^o - \bar{x}^a\| - \frac{1}{2} \right] \right\} - \frac{1}{2} \right] \cos \left( \frac{\pi}{360} \right) \quad (2.21)$$

In addition to relative position, visual stimuli also contains a measurement of the object's identification. The amount of identifying information for players varies inversely to the distance between the observer and the player, allowing the observer to identify a player uniquely, partially, or completely ambiguously:

- If the player is within 20 meters of the observer, the observer perceives complete identification: both the team name and number of the player.

- If the player is between 20 and 40 meters from the observer, the probability that the observer will perceive the player's number decreases linearly from 1 to 0.

\(^7\)All angles in this thesis are measured in radians.

\(^8\)For example, many fielded robots with multiple cameras use knowledge about the pose of each camera along with correlations between individual stimuli to obtain the range and bearing of each captured ray of light [HZ00, EHL+02]. Alternatively, transit time of active light or sound waves can measure range in any particular direction.
• If the player is between 40 and 60 meters from the observer, the observer does not perceive the player's number and the probability that the observer will perceive the player's team name decreases linearly from 1 to 0.

• If the player is further than 60 meters from the observer, then the observer does not perceive any information about the identification of the player.

2.4.3 Internal Timing

SoccerServer simulates continuous dynamics using a discrete algorithm governed by an internal clock. The Clock module produces a sense_body event every 100 milliseconds, and a see event every 150 milliseconds. These events are used to schedule the dynamics simulation and agent stimulation in the following manner:

• Every 100 milliseconds (immediately preceding a sense_body event), the SoccerServer simulates the environment dynamics.

• Every 100 milliseconds (on a sense_body event), the SoccerServer provides proprioceptive stimulus to each agent.

• Every 150 milliseconds (on a see event), the SoccerServer provides visual stimulus to each agent.

Because of the inherent frequency mismatch, three distinct cycle types can be defined: one where the visual stimulus is provided in the beginning of the cycle, one where the visual stimulus is provided in the middle of the cycle, and one where no visual stimulus is provided (see Figure 2.14).

Figure 2.14: Event timing within the SoccerServer module. Cycles types 1 through 3 represent cycles where the visual stimulus is provided in the beginning of the cycle, in the middle of the cycle, and not at all respectively.
2.4.4 Action Failure

The SoccerServer accepts action requests from agents throughout each cycle, but only simulates their effects every 100 milliseconds. Action duration is modelled by the ActionFilter module that restricts each agent to one action in a single cycle; if more than one action is requested, all subsequent requests are ignored — i.e., the action fails (see Table 2.1). It is therefore important that the agent only try to act once per cycle to ensure that all of its action requests are honoured.

Table 2.1: SoccerServer rules for effecting action based on the reception times of sequential action requests. An action request is considered “on time” if it is received before simulation.

<table>
<thead>
<tr>
<th>Previous request</th>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>On time</td>
<td>On time</td>
<td>Effected this cycle</td>
</tr>
<tr>
<td>Late</td>
<td>On time</td>
<td>Not effected (action failure)</td>
</tr>
<tr>
<td>On time</td>
<td>Late</td>
<td>Effected next cycle</td>
</tr>
<tr>
<td>Late</td>
<td>Late</td>
<td>Effected next cycle</td>
</tr>
</tbody>
</table>

2.4.5 Interacting with the SoccerServer

Real agents are situated in a dynamic world and the success of their activity is directly dependent upon the synchronization of sensors, actuators, and deliberation with the environment. This must be achieved in the face of bounded computational resources and unpredictable situations. Although the RSL environment is relatively structured, synchronization has been the topic of much research (see [BPH00] for a survey of popular solutions).

An ideal agent senses the environment, updates its belief in the world state, decides on a course of action based on those beliefs, and effects action all before the environment changes state in such a way as to cause the action or decision to fail⁹. Such behaviour is sufficiently reactive, and can be defined in the RSL with the following temporal precedence constraints on the system.

1. The first constraint is strict: the SoccerServer must receive one action request

⁹One can distinguish between action failures (where the agent attempts to perform the action and fails) and decision failures (where the agent succeeds in action, but the action choice is based on an inconsistent model). Depending on the situation, both can be equally dangerous.
before it begins simulating the dynamics. If the SoccerServer receives the action request after it simulates the dynamics, then the agent will remain idle until the next simulation cycle. If the SoccerServer receives more than one command in a single cycle, the second and any subsequent command is ignored, as described in Table 2.1.

2. The second constraint is soft: the agent should command action only after sufficient sensing and deliberation. If this constraint is violated then the agent is basing its action on less reliable beliefs. For example, the CMUnited98 agent acts at the beginning of each simulation cycle [SVR98]. This virtually ensures the first constraint by abandoning the second: all decisions are based on prediction from the previous cycle and not from current sensor data. Though this method has proved successful in this domain where the dynamics are structured, well known, and easy to simulate, this approach is ill-advised in general.

These constraints can be satisfied by matching the agent's sense-act loop to the environment's natural frequency. In the case of the RSL, an agent must determine the SoccerServer cycle (thereby determining when action is effected, when stimulation is to be expected, and how much deliberation is available) and coordinate with it by using expectations about its deliberation limitations. This is meta-level procedure is called deliberation scheduling [BD94, ZR96, DBS94] and the resulting agent is said to be synchronized with the SoccerServer in that it has determined and adapted to the environment's natural frequency.

Asynchronous operation can result in irrelevant sensing, inconsistent representations, and delayed, failed, or inappropriate action — serious problems which can affect even the most sophisticated skills and plans [BBS+00, Dor00, SS00, Pou00].

The environment's natural frequency can be deduced through sufficient interaction [HPH00, OPS99] but the communication protocol in the RSL is too restrictive to support such standard solutions. The deduction must come instead from accurate system specification [Cri89]. However, simple specifications have been adopted in the past that require unjustified assumptions. For example, two strategies have been described that each exploit different knowledge about the system [BPH00, MM02]:

**Internal synchronization** employs the belief that the SoccerServer cycles at 10 Hz. Under this specification, it suffices to act every 100 ms to satisfy the first constraint of correct action. However, the second constraint is largely overlooked.
**External synchronization** employs the belief that proprioceptive stimulus is provided at the beginning of each cycle. Under this specification (and the assumption of minimal communication delay) the SoccerServer cycle can be inferred and an appropriate action time chosen.

In practise, however, their intended behaviour is complicated by fluctuations in the SoccerServer cycle's periodicity caused by clock inaccuracies\(^{10}\) and fluctuations in perception times caused by varying communication reliability\(^{11}\).

![Timer variation over one minute](image1.png) ![Network transmission variability](image2.png)

**Figure 2.15**: Typical fluctuations in the true and observed SoccerServer cycle measured by timing the production and reception of `sense_body` events respectively. When system resources are strained, the timer provided by its operating system causes the SoccerServer's cycle to fluctuate (left). Additionally, network variation (right) causes the observed cycle to fluctuate as the aggregate of both the timer and network delays.

There exists a theoretical framework for understanding the effect of these complications that shows how internal synchronization, while not very sensitive to network variation, is unlikely to act at the optimal time [MM02]. External synchronization can perform much better because the proprioceptive stimulus can be a good

---

\(^{10}\)The quartz oscillations that computers use to keep time are inaccurate. Typically, a clock can deviate from real time by as much as 30 milliseconds over a 10 minute period [OPS99]. Also, since the system is typically not run over a real-time operating system, interval timers tend to overrun when the processor is overly taxed. These anomalies occur independently across the system which, during the course of a full game, can incur timing deviations that constitute significant portions of the simulation cycle period.

\(^{11}\)Varying network traffic causes unobservable and unpredictable communication delays. Further, since the system uses a lossy communication protocol, messages can arrive out of order or be lost entirely in the transport layer. Therefore, the sensitivity of a synchronization method to such communication faults is extremely relevant.
indicator of a new simulation cycle when communication delays are reliably small. As transmission delay increases, however, the likelihood of satisfying the above constraints decreases rapidly. This is particularly evident with large communication delays where, as shown in Figure 2.16, external synchronization can perform worse than internal synchronization.

![Figure 2.16: A comparison of the Internal (solid) and External (dashed) synchronization algorithms made using the theory introduced in [MM02]. A correct action satisfies both of the constraints described in the beginning of this section.](image)

2.5 Summary

Although an agent may be considered as nothing more than a function from its percepts and history into action, this chapter has expanded on the necessary complexity of such a function for agents in realistic domains. This complexity has hampered development of artificial agents, as each constituent aspect seems to require significant consideration.

- **Perception and modelling**, which combine to manage the agent's internal representation, must adequately describe the state of the world and its inhabitants. This chapter has introduced theoretically acceptable ways of handling vast possibilities left after considering ambiguous percepts using probability theory and approximate techniques.
• *Deliberation* is the process of making goal-oriented decisions based on the agent’s internal representation of the world. Systematic search of the options can solve deliberation with unlimited resources. However, in realistic domains the problem becomes one of optimally allocating the available resources to obtain the best decision possible. This chapter has introduced these issues, and refers to various architectures that address this issue differently.

Ultimately, these constituents must be integrated efficiently into an architecture that allows proactive agents to operate in demanding environments with sufficient reactivity. This chapter has also described the RSL domain, a challenging multiple-agent domain well suited to the study of the facets of agency. The remainder of this thesis will present novel considerations and developments in these areas.
Chapter 3

A Constraint-based Agent Architecture

The typical description of a single physical agent is as the composition of a body (which acts as the physical interface to the agent's environment) and a controller (which polls the body for its interpretations of the environment, and issues commands that evoke action). The precise composition is defined by a particular agent architecture, whose responsibility is to provide certain assurances regarding the agent's properties, such as reactivity, proactivity, taskability, etc..

For example, a human agent has a body, consisting of bones, nerves, muscles, and various organs; and a controller which, although usually considered to be an abstract entity, is believed to reside mainly in the nervous system. The architecture is described, as far as it is understood, by great volumes of anatomy and physiology (and at a finer scale by our genome, chemistry, physics, etc.) along with the whole of psychology and cognitive science.

This chapter presents a novel architecture for an agent situated in a demanding environment, i.e., an environment that requires the agent to make complicated, time-critical decisions. The primary conflict is in attaining rational decisions, which may require significant resources and time, while remaining sufficiently responsive. The conflict is addressed by careful management of the available resources using the intentional stance to reduce the requirements of deliberation, and an adaptive deliberation scheduler that allocates resources in a manner appropriate to the current task and environment.

The mechanism of resource allocation is intimately connected to the fundamental design paradigm of this architecture: each of the controller's particular responsibilities are tackled asynchronously, independently, and in an interruptible manner. The result is an anytime-available action command and the longer the
agent devotes to deliberation — then the more possibilities can be filtered and the more optimized the action command becomes — the more rational the resulting action will be.

The architecture is distinguished by four notable processes organized into CN modules in a layered topology, as shown in Figure 3.1. The precise layer boundaries are not specified, but an operator hierarchy (from the bottom, most reactive component to the top, least reactive component) is inherent.

Figure 3.1: A Constraint Net module of the constraint-based architecture.

- The **Scheduler** module, the most reactive process, monitors the agent’s controller interface to ensure that actions are being effected successfully. By scheduling the command event directly, the Scheduler indirectly controls the resources used by the other modules by controlling how long the agent devotes to deliberation.

- The **Filter** module uses the agent’s percept trace as evidence to make inferences using a probabilistic model of the environment and its inhabitants. These inferences constitute the agent’s internal representation of the external world.

- The **Planner** module uses the inferred state of the environment to reconsider the agent’s set of intentions. Particular intentions can be adopted, abandoned,
or may persist in the set. The combination of adopted intentions then focuses further deliberation.

- The Means-ends reasoning module uses the agent’s current belief about the state of the environment to optimize an action command that suits the adopted set of intentions.

Each of the above modules are elaborated upon separately in the following sections.

3.1 Adaptive Resource Scheduling

Because planning and means-ends reasoning require resources and take time, deliberation as a whole directly opposes the reactivity requirements of situated agents. This opposition can be understood by considering the cost of deliberation and realizing that, at some point, the cost exceeds the benefit in increased decision quality obtained from further deliberation.

As an example, consider a soccer-playing agent who has received the ball and must decide whether to pass or shoot the ball. Soccer is a highly dynamic domain, and the chances are that if the agent devotes too many resources to deliberation (foregoing interpretation of novel percepts, for example) or allows too much time to pass during deliberation (in the attempt to derive an optimal decision) it will find that the quality of the implemented decision is lower than expected. For example, opponents may have used the time to block the pass or shot under consideration.

Clearly, a different definition of optimality is required for decisions; one that directly considers the effect of used time and resources on the quality of a decision. A process that actively considers this type of extension is referred to as a meta-level reasoner because it reasons about when and how to reason.

The architecture described in this chapter relies upon a meta-level reasoner, implemented within the Scheduler module, to achieve the desired responsiveness for the agent. The mechanism of adaptation is the scheduling of the agent’s command frequency. Because of the asynchronous and interruptible operation of the architecture, the scheduling of a command implicitly constrains the resources used by higher layers. The result is that the Scheduler module can select any time to issue the best command that has been collectively designed by the entire controller.

The meta-level reasoning itself is based on an optimistic principle: to deliberate as much as possible, while still producing action commands that can be effected successfully, and are done so in a situation that is still appropriate to the decision. If the meta-level reasoner can determine whether actions are succeeding or failing in
this respect (due to insufficient reactivity, for example) then the Scheduler module can modify the command frequency, thereby affecting the resources available to the other modules, in response. In most cases, this principle induces stable and responsive command scheduling that is able to adapt to the evolving demands of the agent's task and environment.

3.2 Interpreting Observations

An agent is a function from sensation into action, albeit a complicated function. Internally, stimulus from the environment is converted into percepts by the agent's sensors, before being interpreted by the agent controller. It is up to the agent to then reconstruct the likely state of the environment from its perceptual data. The reconstruction is the agent's internal representation, and is used to solve problems and make decisions concerning the real world.

The internal representation used in this architecture is based on a probabilistic model of the agent's domain: a collection of random variables that collectively describe the relevant state of the environment. The model is derived by choosing the constituent variables as well as specifying their dependencies upon each other and time. For example, Section 2.3 describes a number of typical models, and a new model that considers the behaviours of multiple, observed agents using the intentional stance is described in Chapter 4.

Percepts are evidence about the true state of the environment, but can be ambiguous, incomplete, and noisy. Nonetheless, the agent is able to use the probabilistic model to infer the likelihood of different possibilities based on this evidence as described in Section 2.2. In this way, the Filter module maintains consistent beliefs about the environment using a Bayesian filter and its particular model. The distribution over world states is then used by the agent to influence its decision process.

3.3 Deliberation

Deliberation is an agent operation wherein the internal representation is considered to arrive at a decision concerning action. Within the architecture described in this chapter, deliberation is performed concurrently by two separate processes: planning and means-ends reasoning.

The highest level of deliberation is planning: the consideration, evocation, and revision of intentions. This process, carried out in the Planner module, is fashioned on the intentional stance view of agency where intentions are represented
as appropriate constraints (formally, as a list of conjunctions of constraints) that are specifically designed to achieve some proposition or goal. Once the agent has committed to an intention, the agent pursues it until it concludes that it has been satisfied or that it will never be satisfied and should be abandoned. The agent then only considers options that are consistent with its currently adopted intentions, reducing decision complexity.

The lower level of deliberation is means-ends reasoning: the selection of specific action designed to satisfy the agent’s intentions. Performed in the Means-ends reasoning module, this process optimizes an action command to achieve the intended set of constraints. The optimization is a gradient ascent traversal of the world’s state-space, performed using available actions, into regions of satisfiability.

The list of conjunctions of constraints that specify an intention can be viewed as a plan where each list element is an abstract action designed to achieve a particular task. However, this is more general than an ordered sequence of action commands: in this scheme, individual action commands are constructed online based on the success of previous actions and the believed state of the environment, while the agent actively considers alternate options. In this way, the intentions aggregate with a constraint solver to define the agent’s particular policy.

3.4 Implementation

A constraint-based agent architecture has been introduced in general. In order to design a functional agent controller based on this architecture, however, the operation of many of the processes described in the previous sections must be specified in more detail.

Chapter 4 introduces a specific model of a multiple-agent environment that can be used in the architecture, particularly by the Filter module; and Chapter 5 will provide the specifics required to implement this agent for the RoboCup Simulation League domain. Ultimately, Chapter 6 will explore whether the combination of reactive and deliberative components, under control of the resource scheduling meta-reasoner, are able to successfully operate in a highly dynamic and interactive environment.
Chapter 4

Modelling Environment and Agent Behaviours

This chapter is concerned with empowering an observer to understand the dynamics of an inhabited environment. The observer may be a passive witness who analyzes percepts from video or some other sensor without impacting the environment; or be an embedded actor who uses its analysis to influence the system. However, the observer is assumed to exist independently from the observed agents in terms of its goals and actions (i.e., the observer is not a member of the observed group). The actual use of such empowerment will not be addressed at all in this chapter, except to posit that it is in fact useful, for example, to better understand a group of other agents' actions.

To “understand” the dynamics of an inhabited environment, the observer can compare percepts with a probabilistic model of these dynamics and infer various hidden properties. In this way, perception evolves beyond direct sensory stimulation to be the dynamic inference of useful characteristics. This chapter will introduce a specific model that enables an observer to better interpret a multiple-agent environment. This model is shown in Figure 4.1.

Previous work has shown that a set of variables' dynamic evolution can be theoretically modelled using a hidden Markov model (HMM). In the model described in this chapter, the HMM structure is borrowed to model the fundamental elements of the world state, such as the position and velocity of objects. While this seems to work well for such simple vectors with well defined dynamics, enlarging the vector

---

1Although an agent is defined as an entity that acts in its environment, the robustness and appropriateness of that action is intimately connected with the evolving state of the environment and its co-inhabitants. Competent analysis of these dynamics can be an important component of a decision process. However, the model's full application in a decision process is not explored in this thesis.
to model complexities that arise out of the exogenous actions of other agents, for example, makes learning and inference in the HMM intractable or inaccurate.

Therefore, to postulate other information the model must be extended with more restrictive dependencies. This extension is based on the hypothesis that observed agents are intentional — i.e., possess beliefs, desires, and intentions in accordance with the psychological theory of intentionality and whose inference serves as an abstract behavioural description. This structured elaboration imposes a particular independence topology on the modelled variables, partially shown in Figure 4.1 and further elaborated upon throughout this chapter.

The underlying motivation of this model as a whole is the assumption that abstraction of an observed agent’s policy to the facets of intentionality (those described in Section 2.1.2) will attain computational tractability while retaining an appropriate amount of modelling accuracy.

The remainder of this chapter will discuss the random variables used in this model along with their dependencies. Section 4.5 develops an inference algorithm for the model, and theoretically analyzes the inference complexity. In Section 4.6, a method of predicting model parameters making use of particular observations is provided based on the Expectation Maximization algorithm.

4.1 Agent Beliefs

Intentional agents base their decisions on their own beliefs — an internal representation of the world state, constructed from particular perception and prior knowledge. The model in Figure 4.1, as a whole, constitutes the observer's beliefs: it is a probabilistic internal representation of the state of the world, including the mental attributes of a group of intentional agents (excluding the observer). The model assumes that these observed agents are intentional themselves and, hence, possess beliefs of their own. This is modelled by the random variables \( B_t \) which are the observer’s beliefs about the observed agents’ beliefs.

Agents from the same group are assumed to have the same perceptual process (the mapping of percepts to belief about the external environment). Of course, these agents do not have the same beliefs, as they each hold a different perspective in the environment. In general, agent beliefs may be logically inconsistent both internally (among components of the belief) and externally (with respect to other agents or the true world state). However, the observer should assume some range of acuity in the observed agents’ perceptual processes in order to simplify parameter estimation and inference. In the RSL, each particular world state provides deterministic percepts to each agent (percepts are ambiguous but are not stochastic). A
Figure 4.1: A hierarchical, probabilistic model of multiple, intentional agents acting in the world, represented as a dynamic belief network. The bottom of the figure may be recognized as a hidden Markov model of the world state. The world state is dependent at each time step by its history as well as the actions, $A_i$, of all $p$ agents. How each agent arrives at an action is based on the theory of intentionality: each agent has an individual intention, $I_i$, and the group as a whole has a joint intention, $J$. Further, each agent has a belief about the state of the world, $B_i$, which is used to determine the satisfaction or abandonment of individual and joint intentions ($S_i$ and $T_i$ respectively) as well as their adoption.

A simple mapping from this percept is a reasonable lower bound on the perceptual acuity. Alternatively, an agent may filter its percepts (for example with a Kalman filter or another approximate Bayesian filter) with increasing accuracy up to the optimal filter.

An acuity range is enforced in the model by specifying a set of logical propositions concerning the environment, and allowing agents to bet on their truth values. The observer can then learn the similarities in how its beliefs about the environment ($X$) are formed and those of other agents ($B$). For the RSL, appropriate environmental propositions include the following:
a discretized position of each agent,
• distances between opponents,
• distances between opponents and their nearest teammate,
• a measure of the ball’s interceptability if passed from the ball carrier to each of its teammates, and
• a measure of each opponent’s ability to score with a shot to the goal.

For other domains, beliefs may include other propositions, even modelling higher-order beliefs (e.g., beliefs about beliefs). Given a set of propositions, the correlation between an agent’s beliefs and the observer’s beliefs can be parameterized by a number of probabilities (one less than the number of propositions) under the assumption that each proposition is independent of each other and that the correlation is independent of the proposition’s value. These probabilities, parameterized within the vector $\vec{\beta}$, fully specify the observer’s model of perceptual difference between itself and the observed agents; as is described in Equation 4.1.

$$
\Pr(B_q^T = \bar{b} | X_t = \bar{x}) = \prod_{i=1}^{||B||} \begin{cases} 
\beta_i & \text{if } b_i \leftrightarrow x_i \\
1 - \beta_i & \text{else}
\end{cases}
$$

(4.1)

### 4.2 Agent Intentions

The elaborated model is centred around the assumed intentionality of observed agents. One salient feature of this intentionality is that both individuals and groups commit to bringing about particular propositions (under a host of assumptions and constraints imposed by the theory of intentional systems, introduced in Section 2.1.2). Therefore, individual intentions are assumed for each observed agent $q$, and are modelled using the multinomial random variables $I_q^T$, whose domain covers the commitments to all possible such propositions. The model assumes that each observed agent has exactly one intention. Although the details of the intended proposition may be unknown, its ultimate value is directly modelled using the boolean random variables $S_q^T$, which are true when the propositions are true or will never be true. The remainder of this section describes the dynamics of intentions, as modelled using these two variables: Section 4.2.1 describes how intentions are adopted and how they evolve; and Section 4.2.2 describes how intentions are dropped.

When observing and trying to understand a group of intentional agents, as opposed to an individual, the group as a whole is similarly assumed to have a joint
intention. Joint intentions are handled similarly to individual intentions by the analogous random variables $J_t$ and $T_t$, which model the group's joint intention and whether it is (or never will be) satisfied respectively. However, joint intentions are hierarchically superior to individual intentions in that the group's joint intention can influence that of an individual. These specifics are detailed in the remainder of this section.

4.2.1 Intent Formation

Although agents may intend to satisfy many propositions, all of these must be logically consistent. The multinomial intention random variables, $I^q_t$, are indexes all possible sets of logically consistent propositions. Therefore, this unitary variable can be used to model multiple, concurrent intentions to unclarified propositions. Equivalently, $J_t$ indexes into propositions that are used as joint intentions of the observed group.

As is described in Section 4.2.2, an agent's intention persists until it is satisfied or will never be satisfied. This is a different mechanism as used by the hierarchical hidden Markov model described in Section 2.3.5, which allows more general stochastic transitions. However, the restriction simplifies parameter estimation and produces more accurate inference, if appropriate.

The adoption of a group's joint intention requires mutual desire in a particular proposition and mutual belief that the desired proposition is not satisfied. This requirement is modelled within the mapping $J : B^{1:p} \rightarrow J$, which describes the distribution of likely joint intentions that will be adopted given a certain set of the group members' individual beliefs.

$$J(b_1:p) = \Pr(J_{t+1} | T_t = T, B_t^{1:p} = b_1:p)$$

The distribution $J$ performs a vital role in the joint intention's dynamics, which are modelled by the conditional probability function in Equation 4.3.

$$\Pr(J_{t+1} = j' | J_t = j, T_t = \tau, B_t^{1:p} = \bar{b}_1:p) = \begin{cases} J^{(j')} (\bar{b}_1:p) & \text{if } \tau \\ \delta_k (j' - j) & \text{else} \end{cases}$$

Individual intentions obey similar dynamics, except that their adoption is based on the agent's beliefs about the state of the world and the group's joint intention. This process is described by the intent formation models (parameterized by the mappings $I_j : B \rightarrow I$, one for each possible joint intention) which describes the initial distribution of intentions given particular joint intentions and beliefs.
Based on this distribution, individual intention dynamics are modelled by the conditional probability function in Equation 4.5.

$$\Pr(I_{t+1} = i' | B_t = \bar{b}, I_t = i, S_t = s, J_{t+1} = j) = \begin{cases} f^{(i')}(\bar{b}) & \text{if } s \\ \delta_k(i' - i) & \text{else} \end{cases}$$

The modelled dynamics of joint and individual intentions exhibit context-specific dependence and the values of $S_t$ and $T_t$ can be viewed as topology selectors of the conditional dependencies. For example, the dynamics of individual intentions can be viewed in two simple cases indexed by the value of $S_t$, as seen in Figure 4.2. This feature will be a significant factor in developing practical inference algorithms.

4.2.2 Intent Abandonment

Individual intentions are abandoned when the proposition that the agent has committed to becomes true or the agent determines that the proposition can never become true. Sometimes this is contingent upon the group’s joint intention, so that an agent can choose to terminate its individual intention if the joint commitment is dropped. This allows more general dynamics than hierarchical HMMs and abstract HMMs, who only allow higher-level HMMs to transition after lower-level HMMs terminate.
Individual intention persistence is modelled by the random variables $S_i^\theta$. Specifically, intentions persist while $S_i^\theta = \top$, and are relinquished when $S_i^\theta = \bot$. The agent's decision to abandon a particular intention is modelled by the mapping $S_i(b, \tau)$, that specifies the probability that a particular intention is satisfied if the agent has particular beliefs.

$$S_i(b, \tau) = \Pr(S_i^\theta = \top | I_i^\theta = i, T_t = \tau, B_i^\theta = b)$$  \hspace{1cm} (4.6)

The differences between $S_i(b, \top)$ and $S_i(b, \bot)$ characterize the agent's commitment to the group: conforming agents manage their individual intentions in direct correspondence to the group's joint intention ($S_i(b, \top) = 1$) while independent agents disregard the state of the group's joint intention entirely ($S_i(b, \top) = S_i(b, \bot)$).

A group's abandonment of its joint intention requires mutual belief that the desired proposition is satisfied or will never be satisfied. This condition is modelled through the boolean random variables $T_t$ and the mapping $T_j(b_1, p)$, that specifies the probability that a group drops their joint intention given a particular set of beliefs.

$$T_j(b_1, p) = \Pr(T_t = \top | J_t = j, B_i^{1p} = b_1, p)$$  \hspace{1cm} (4.7)

### 4.3 Agent Desires

A notable exception to the explicitly modelled attributes is desire. Desire is the prioritization of options without consistency constraints, prior to commitment. The model does assume that agents have desires, but they are modelled implicitly as influences of the intent formation model. In effect, desires are integrated out during the model parameter estimation process.

### 4.4 Agent Action

An agent's actions can only be perceived through changes in the steady-state dynamics of the environments. These changes are interpreted using an action model that describes how actions effect the environment. For example, there are three fundamental actions in the RSL domain:

**Dash** is an action that applies a force to the agent in the direction of its current heading. The Dash action effects only the player's position and velocity in accordance with the RSL equations of dynamics (Equations 2.17 and 2.18) with $\vec{f} = m a \leq \theta^p$, where $a$ is the power parameter of the Dash command.
Turn is an action that applies an instantaneous angular moment to the agent that modifies its heading, $\theta^p$

Kick is an action that applies a force to the ball (if the ball is sufficiently close to the kicker). It is sometimes useful to specialize the Kick action. An observer may consider the Shot action (a Kick action which gives the ball a trajectory into the goal) the Pass action (a Kick action which gives to ball a trajectory towards another teammate) and the SelfPass action (a Pass to itself — i.e., to a location where the kicker is able to intercept the ball before any other agent).

How an agent chooses an action (i.e., its policies) is parameterized by the mappings $A_i$, one for each possible individual intention.

$$A_i(\bar{b}) = \Pr(A_t^{q_x} I_t^{x+1} = i, B_t^{q} = \bar{b}) \quad (4.8)$$

4.5 Inference

The purpose of inference is to reason about elements of the world that cannot be observed, but are hypothesized in a particular model of the world (for example, the probabilistic model described in the previous sections of this chapter). The RSL agent will perform two inference tasks on this model, both of which are sub-classes of general Bayesian inference in dynamic belief networks:

1. Filtering sequential observations to infer marginal distributions over variables that are useful influencers of its decision process. For notational purposes, the set of modelled, hidden random variables can be referred to singularly,

$$Q_t \equiv \{J_t, I_t^{1:p}, A_t^{1:p}, X_t, B_t^{1:p}, S_t^{1:p}, T_t\} \quad (4.9)$$

Then forward inference, or filtering, is generally characterized by the calculation of the distribution $\Pr(Q_{t+1} | Y_{0:t})$. This task is described in Section 4.5.1.

2. Inferring the most likely distribution across the entire network (i.e., for each time slice) given an entire sequence of observations. Generally, this inference is characterized by the calculation of the distribution $\Pr(Q_t | Y_{0:T} = \bar{y}_{0:T})$. This task is described in Section 4.5.2, and is required to estimate the parameters of the model from evidence.
4.5.1 Forward Inference

Forward inference, also called filtering, can be defined as the iterative computation of the filtering distribution \( \Pr(Q_{t+1}|Y_{0:t}) \). The general prediction and updating steps of Bayesian filtering (described in Section 2.2.1) can be used to perform this calculation recursively:

1. The first step is the prediction of the initial filtering distribution which is defined in the semantics of the single time-step belief network as the parameter \( \pi \).
2. The agent will then compute a posterior for the current time step based on an observation who influences the posterior as in Equation 4.10.

\[
\Pr(Q_t|Y_{0:t}) = \frac{\Pr(Q_t|Y_{0:t-1}) \Pr(Y_t|X_t)}{\Pr(Y_t|Y_{0:t-1})}
\]  

(4.10)

3. The agent will then predict a prior over the filtering distribution in the next time step, making use of the fact that future time steps are independent of past time steps given the current time step:

\[
\Pr(Q_{t+1}|Y_{0:t}) = \int \cdots \int \Pr(Q_t|Y_{0:t}) \Pr(Q_{t+1}|Q_t)
\]  

(4.11)

4. Steps (2) and (3) are then repeated indefinitely as long as observations are made (i.e., \( \forall t \in [0, T] \)). Afterwords, future predictions can be made by repeatedly applying Equation 4.11 with \( Y_{0:t} \) replaced with \( Y_{0:T} \).

In general, it is important that the agent make use of both the primary structure (the conditional dependencies portrayed by the belief network) as well as the secondary structure (the relational dependencies contained within the conditional probability functions themselves) during inference. This becomes especially important during the prediction step, step (2), since intentions persist until they are satisfied. Therefore, for possible worlds where \( T_t \) or \( S_i^t \) are true the corresponding joint intention or individual intention are simply copied to the new time step; for possible worlds where they are false, the new joint or individual intention depends on a reduced set of variables (see Figure 4.3).

Forward inference has space complexity of \( O(||Q|| \times T) \), and time complexity of \( O(||Q||^2 \times T) \), where \( ||Q|| \) is the size of the joint distribution’s domain. For the this model,

\[
||Q|| = ||J|| \cdot ||X|| \cdot (||I|| \cdot ||A||)^p \cdot 2^{p(||\beta||+1)+1}
\]  

(4.12)
4.5.2 Backward Inference

The other required inference is the computation of the backward distributions $\Pr(Y_{t+1:T}|Q_t)$, which combine with the product of forward inference to perform the second inference task:

$$\Pr(Q_t|Y_{0:T}) = \frac{\Pr(Q_t|Y_{0:t}) \Pr(Y_{t+1:T}|Q_t)}{\Pr(Y_{t+1:T}|Y_{0:t})} \quad (4.13)$$

Given an observation sequence, these distributions can be computed with the following iterative computation:

1. The first step is the prediction of the backward distribution for the last time step of the observed sequence:

$$\Pr(Y_T|Q_{T-1}) = \int \cdots \int \Pr(Q_T|Q_{T-1}) \Pr(Y_T|X_T) \quad (4.14)$$

2. Given this base, the agent can then recurse back through sequence using Equation 4.15 to compute the remaining backward distributions (i.e., $\forall t \in [0, T-2]$).
\[ \Pr(Y_{t+1:T}|Q_t) = \int \ldots \int \Pr(Q_{t+1}|Q_t) \Pr(Y_{t+1}|X_{t+1}) \times \Pr(Y_{t+2:T}|Q_{t+1}) \]  

(4.15)

Backward inference has space complexity of \( O(\|Q\| \times T) \), and time complexity of \( O(\|Q\|^2 \times T) \), in addition to the computation of the forward distributions.

If a joint distribution over more than one time step is desired, the inferrer can simply skip the marginalization in Equation 4.15.

\[ \Pr(Q_t, Q_{t+1}|Y_{0:T}) = \frac{\Pr(Q_t|Y_{0:t}) \Pr(Q_{t+1}|Q_t) \Pr(Y_{t+1}|Q_{t+1}) \Pr(Y_{t+2:T}|Q_{t+1})}{\Pr(Y_{t+1:T}|Y_{0:t})} \]

(4.16)

### 4.5.3 Approximate Inference

Closed-form Bayesian inference in model of Figure 4.1 can be intractable, depending on the size of each variable's domain and the number of modelled agents. Therefore, a situated agent requires an approximate algorithm that can exploit local structures and achieve computational practicality, while retaining sufficient accuracy under uncertainty. In general, hybrid inference algorithms are developed by identifying a minimal marginalization of the model that allows exact inference. If a suitable marginalization can be found, then an algorithm can be developed that performs a stochastic search in a subset of conditions, which is assumed to be sufficiently representative of all possible worlds [DdFMR00].

Such a marginalization can be devised by exploring the chain

\[ C_t \overset{\text{def}}{=} \{J_t, I_t^{1:p}, A_t^{1:p}, X_t\} \]

The chain itself has nice inference properties both within each time step (where every node is conditioned upon no more than one other random variable) and between time steps (where transitional probability functions are sparse for most probable situations). Also, the chain d-separates the evidence node from the remainder of the joint distribution \( (\Psi_t \overset{\text{def}}{=} Q_t - C_t) \); in other words, \( \Psi_t \) is independent of the observations given the chain as shown in Figure 4.4.

This segregation can be exploited by the inference algorithm. Because of the separation of \( \Psi_t \) from observations, the posterior step in forward inference need only be computed for the chain \( C_t \):

\[ \Pr(C_t|Y_{0:t}) = \frac{\Pr(C_t|Y_{0:t-1}) \Pr(Y_t|X_t)}{\Pr(Y_{t}|Y_{0:t-1})} \]

(4.17)
 Likewise, the prediction step becomes:

\[
\Pr(C_{t+1}|Y_{0:t}) = \int \ldots \int \Pr(C_t|Y_{0:t}) \int \ldots \int \Pr(\Psi_t|C_t) \Pr(C_{t+1}|Q_t) \quad (4.18)
\]

This computation is no less complex than before the separation of \(Q_t\) into \(C_f\) and \(\Psi_t\), but Equation 4.18 does present a promising opportunity for a stochastic search approximation. Instead of considering its entire domain, only select possibilities of the value of \(\Psi_t\) can be used as an approximation of the entire space. If these select possibilities are chosen appropriately (for example, by the processes described in Section 2.2.1.1) the approximation is likely to produce reasonable inferences.

Specifically, the agent can consider \(N_\psi\) possible values of \(\Psi_t\) producing the approximation in Equation 4.19. Using this approximation, forward inference has space complexity \(O(||C||T + N_\psi T)\) and time complexity \(O(||C|| \times N_\psi \times T)\), where

\[
||C|| = ||J|| \cdot (||I|| \cdot ||A||)^P \cdot ||X||
\]

— a significant reduction (by a factor of \(N_\psi \times 2^{-P(||\beta||+1)-1}\)) as long as sufficient accuracy can be achieved with a conservative number of samples.

\[
\tilde{\Pr}(C_{t+1}|Y_{0:t}; N_\psi) = \int \ldots \int \Pr(C_t|Y_{0:t}) \times \\
\sum_{i=0}^{N_\psi} \Pr(\Psi_t = \tilde{\psi}_t^{(i)}|C_t) \Pr(C_{t+1}|C_t, \Psi_t = \tilde{\psi}_t^{(i)}) \quad (4.19)
\]

Despite this improvement, the size of the domain of \(C_t\) is still troublesome for one reason: \(X_t\) is a continuous random variable in a large space. Therefore, the
agent must also sample possible world states. If the agent considers \( N_x \) possible world states, and \( C_t^{-X} \text{def} C_t - \{X_t\} \), then this can be done using Equation 4.20.

\[
\bar{\Pr}(C_{t+1}|Y_{0:t}; N_\psi, N_x) = \int \ldots \int_{C_t^{-X}} \sum_{i=0}^{N_x} \Pr(C_t^{-X}, X_t = \bar{x}_t^{(i)}|Y_{0:t}) \times \\
\sum_{j=0}^{N_\psi} \Pr(\Psi_t = \bar{\psi}_t^{(j)}|C_t^{-X}, X_t = \bar{x}_t^{(i)}) \times \\
\Pr(C_{t+1}|C_t^{-X}, X_t = \bar{x}_t^{(i)}, \Psi_t = \bar{\psi}_t^{(j)})
\] (4.20)

Filtering \( \bar{\Pr}(C_{t+1}|Y_{0:t}; N_\psi, N_x) \) requires \( O((||C^{-X}|| + N_x + N_\psi) \times T) \) space and \( O(||C^{-X}|| \times N_x \times N_\psi \times T) \) time where \( ||C^{-X}|| = ||J|| \cdot (||I|| \cdot ||A||)^p \).

4.5.3.1 Representing approximate distributions

Monte Carlo approximations are characterized by sampled location-weight pairs; for example, in the space of \( \Psi \) and \( X \). In the RSL domain there are also informative bounds on portions of the domain that have non-zero probability. For example, the bounds on probable portions of an observed object’s position is given by the maximum errors in relative range and bearing in Equation 5.10. Such bounds on the true distribution are also maintained in the representation of the model’s distributions; and are used to distinguish between unlikely and impossible locations and to guide sampling (helping to overcome the problem of lost, improbable hypotheses).

If these bounds ever become convex, then the forward inference procedure will retain that property. Because convex bounds can be maintained efficiently,\(^2\) the perception bounds are represented as the minimal trapezoid that covers the true bounds, as shown in Figure 4.5. With this approximation, the bounds of any filtered distribution will also be convex and will always over-estimate the true bound. In general, the reduced confidence caused by the over-estimate is outweighed by the benefits of the representation. The minimal amount of over-approximation means that it is extremely unlikely that a randomly selected location will lie outside the true distribution.

\(^2\)If \( n \) and \( m \) are the number of vertices in two polygons, assuming convexity reduces intersection complexity from \( O(nm) \) to \( O(n + m) \) [Sha78] and sample testing from \( O(n) \) to \( O(\log_2 n) \).
Figure 4.5: Observation distributions take the shape of a ring sector due to ambiguity in both range and bearing (bounded by the solid line). The ring sector is represented using a minimal trapezoid that covers the true area (bounded by the dashed line). The shaded area represents portions of the representation not included in the true bounds (but, note that the curvature of the arcs have been exaggerated in this figure).

### 4.6 Parameter Estimation

A probabilistic model can be described with three components: a list of modelled random variables (for example, the semi-infinite set $Q_t$) a factorization of their joint probability distribution (which can be captured within a graphical representation like that in Figure 4.1) and conditional probability functions characterizing the relationships implied in the graphical representation (described and parameterized in the previous sections).

The intentional model described in this chapter has the following parameters which describe the conditional probability functions, or secondary structure, of the model:

\[
\Theta = \langle \pi, J, I_j, A_i, I_j, S_i, T \rangle
\]  

(4.21)

- The prior, joint distribution over the modelled variables, $\pi \overset{\text{def}}{=} \Pr(Q_0)$;
- The mapping $J$, representable by a $(||J|| - 1) \times 2^{||\bar{\beta}||}$ matrix, that parameterizes the group’s commitment to particular joint intentions;
- The $||J||$ mappings $I_j$, each representable by a $(||I|| - 1) \times 2^{||\bar{\beta}||}$ matrix, that parameterizes each agent’s commitment to an individual intention;
An approximate to the filtered distribution of agent positions, \( \text{Pr}(X_t^a | Y_{0:t}) \). The dark shaded polygon represents all possible positions. Circles represent the location (centres) and posterior value (radii) of 20 samples.

Predicting the agent’s next position (\( \text{Pr}(X_{t+1}^a | X_t^a, A_t^a, Y_{0:t}) \), shown below) is based on the above prior (whose bounds are shown in dashed lines) and the force that is applied to the agent. Samples diverge and re-normalize in accordance to the dynamics’ stochasticity.

The predicted distribution (dashed) is then validated through observation modelled by the perception probability distribution (shaded).

To determine the new filtered distribution, \( \text{Pr}(X_{t+1}^a | Y_{0:t+1}) \), the predicted and perception bounds are intersected and sample weights are multiplied by the perception probability at their location. The weights of four delta functions go to zero, and are resampled.

Figure 4.6: An example of Bayesian filtering the world state. The prediction and update steps of the Bayesian filter are outlined using this representation in Figure 4.6, although, remember that no visual stimulus is perceived by the agent every third cycle.
The $|I|$ mappings $A_i$, each representable as a $|A| \times 2^{|I|}$ matrix, that parameterizes each agent's action policy;

- The vector $\bar{\beta}$, that parameterizes the correlation between the observed agent's perceptual acuity and the observers;

- The $|I|$ mappings $S_i$, representable by a $|I| \times 2^{|I|}$ matrix, that parameterizes the conditions under which individual intentions are satisfied; and

- The mapping $T$, representable by a $|J| \times 2^{|I|}$ matrix, that parameterizes the conditions under which group intentions are satisfied.

Specifying these parameters accurately is extremely difficult to do, so it is preferable to learn them from observations of the actual system. For example, the agent can fit the model parameters to percept traces of particular agents that it has stored from previous experience. The remainder of this section explores this concept, and discusses the Expectation Maximization algorithm (introduced in Section 2.2.2) as a method of parameter fitting. The development starts with the familiar temporal segregation of the EM iteration equation into prior and transitional components:

$$
\theta_{k+1} = \arg\max_{\theta} \left( \sum_{Q_0 : T_i} \Pr(Q_0 : T_i | Y_0 : T_i = \bar{y}_0 : T_i ; \theta_k) \times \right.
\left. \log \Pr(Q_0 ; \theta) + \sum_{t=1}^{T_i} \log \Pr(Q_t | Q_{t-1} ; \theta) + \sum_{t=0}^{T_i} \log \Pr(Y_t | Q_t) \right)
$$

(4.22)

The motivation is that the set of parameters that maximize Equation 4.22 can be found by considering the maximization of each summed term independently. For example, maximization of the prior parameter, $\pi = \Pr(Q_0 ; \theta)$, can be performed by considering the first term only, as is done in Equation 4.23.

$$
\pi_{k+1} = \arg\max_{\pi} \left( \sum_{Q_0} \Pr(Q_0 | Y_0 : T_i = \bar{y}_0 : T_i ; \theta_k) \log \pi + c(Q_t) \right)
$$

(4.23)

$$
= \arg\max_{\pi} f(\pi)
$$

(4.24)

where $c$ is constant with respect to $\pi$.

The appropriate parameter can be found using the Lagrangian multiplier technique of minimization. Making use of the fact that, for example, the prior is constrained by the axioms of probability, a Lagrangian constraint can be defined as
\[ \varphi(\pi) = \int \ldots \int \pi^{(q)} \, dq - 1 = 0 \] (4.25)

then, the value of \( \pi \) that maximizes \( f - \lambda \varphi \) will also maximize \( f \); and will serve as the next iterate’s prior parameter.

\[
\frac{\partial}{\partial \pi} \left\{ f - \lambda \varphi \right\}_{\pi = \pi_{k+1}} = 0
\]

\( \pi_{k+1} = \Pr(Q_0|Y_{0:T}; \theta_k) \)

The remaining ‘transitional’ terms of Equation 4.22 can be decomposed so that each term contains only one parameter to maximize. This is done for the intentional model in Equation 4.27.

\[
\sum_{t=1}^{T} \log \Pr(Q_t|Q_{t-1}; \theta) = \sum_{t=1}^{T} \log \Pr(J_t|J_{t-1}, B_{t-1}^{1:p}, T_{t-1}; J) + \log \Pr(X_t|A_t^{1:p}, X_{t-1}) + \log \Pr(Y_t|X_t) + \sum_{q=1}^{p} \log \Pr(I_t^q|J_t, I_{t-1}^q, B_{t-1}^q, S_{t-1}^q; I) + \log \Pr(A_t^q|I_t^q, B_{t-1}^q; A) + \log \Pr(B_t^q|X_t; \bar{\theta}) + \log \Pr(S_t^q|I_t^q, B_t^q, T_t; S)
\]

Each term in the sum is associated with a single parameter. After Lagrangian maximization with respect to each parameter, the following parameter update equations are obtained.

\[
\mathcal{J}(\bar{b}_{1:p}) \leftarrow \frac{\sum_{t=0}^{T-1} \Pr(J_{t+1}, T_t = T, B_{t}^{1:p} = \bar{b}_{1:p}|Y_{0:T})}{\sum_{t=0}^{T-1} \Pr(T_t = T, B_{t}^{1:p} = \bar{b}_{1:p}|Y_{0:T})}
\]

\[
\mathcal{I}_j(\bar{b}) \leftarrow \frac{\sum_{t=0}^{T-1} \sum_{q=1}^{p} \Pr(I_{t+1}^q, J_{t+1} = j, S_t^q = T, B_t^q = \bar{b}|Y_{0:T})}{\sum_{t=0}^{T-1} \sum_{q=1}^{p} \Pr(J_{t+1} = j, S_t^q = T, B_t^q = \bar{b}|Y_{0:T})}
\]

56
Each of these computations is discussed in Section 4.5. The resulting EM algorithm for this model is then

1. to choose initial parameters,\(^3\) and then

2. to repeatedly update the parameters (using each sequence in the data set with Equations 4.26, and 4.28–4.33) until the parameters have converged.

### 4.7 How to Use the Model

The model presented in this chapter can be used by any observer who wishes to infer, from a trace of percepts, the past, present, or future properties of the environment or of other, observed agents. For example, the model can be used by a security system, video annotator, or an agent interacting with the observed agents. Depending on the nature of its use, different inference algorithms, including varying approximations,

\(^3\)The most critical consideration of the EM algorithm is the initial parameters upon which to operate. In complicated environments, the likelihood surface is inherently complex with many local extrema. This means that a hill climbing algorithm like EM is vulnerable to learning globally sub-optimal parameters.
may be used. These algorithms possess different computational requirements and provide different levels of accuracy.

However, the foremost concern with regards to accuracy is the model itself: whether the assumptions of intentionality within the observed agents is valid, and whether the model parameters themselves are applicable to the observed agents. Parameterization provides an infinite number of models describing different groups of intentional agents. The correct parameters can be guessed at, or be computed based on experience in observing the agents of interest, as described in this chapter.

The model can then be used to infer probabilistic distributions over a group of agent's actions, beliefs, individual intentions, and joint intentions. These inferred distributions, which represent the observer's beliefs about the state of the world, can be used in any number of ways from a historic log of events that can be used to learn global properties of the system, to direct influences of the observer's action. As previously stated, these applications are not fully explored in this thesis and are left for future work. In the next chapter, however, the model will be applied to an RSL agent where these distributions constitute an internal representation of the external world. Chapter 5 will also detail how particular elements of the representation might be used as decision influences, but stops short of incorporating the entire internal representation including the assumed intentional properties of observed agents.
Chapter 5

Application to the RoboCup Simulation League

Chapter 3 and 4 have introduced, in general terms, a novel agent architecture and a probabilistic model of a multiple-agent environment, respectively. However, to understand these general concepts further, and to explore the properties of their operation, the theories must be implemented and tested. This chapter will detail the development of an agent for the RoboCup Simulation League (RSL) domain that is designed using the constraint-based architecture of Chapter 3 with the model of Chapter 4 providing internal interpretations of its percepts.

The RSL system, shown partially in Figure 5.1, is composed of multiple, distinct software programs that interact through a lossy communication channel: the Agent programs that implement the agent controllers for 22 agents, and the SoccerServer program that implements the simulated environment including the ball and referee. The SoccerServer was described in Section 2.4. The communication channel can be modelled as a CN module which introduces an exponentially distributed transmission delay that is independent of time [Bol93]. Specifically, for each input event trace, $e(t)$, a transduction produces the event trace $e(t - \Delta)$, where $\Delta$ is a random variable distributed according to Equation 5.1.

\[
\Pr(\Delta = \delta) = \begin{cases} 
\frac{1}{\mathcal{E}[\Delta]} \exp \left( \frac{-\delta}{\mathcal{E}[\Delta]} \right) & \text{if } \delta \geq 0, \\
0 & \text{if } \delta < 0.
\end{cases}
\]  

(5.1)

Although an RSL agent has no physical body, there is still a notion of sensors and actuators in a virtual body connecting a controller to the environment. Further, the simulated environment is demanding and adequately models failures and noise in sensors and actuators. Therefore, all of the agency issues discussed in the preceding chapters apply. The following sections will detail each aspect of the agent.
5.1 Adaptive Resource Scheduling

Any agent-environment system can be characterized by a time-dependent (usually monotonically increasing) utility function that describes the trade off between deliberation cost and decision quality. In the RSL, there is no deliberation cost until the SoccerServer simulates the current time step, where the deliberation cost increases to some fixed positive value. Successful situated agents must use this function to trade off decision quality with cost [Sim82, RW91].

The Schedule module, whose internal structure is shown in Figure 5.2, is responsible for providing prompt, reliable action in a dynamic and adapting environment. The scheduler is a meta-reasoner who decides how long to deliberate before commanding an action based on the Dynamic Action Index algorithm [MM02]. The Dynamic Action Index algorithm uses an interval timer, Internal Clock, as an internal representation of the SoccerServer’s cycle and actively tries to synchronize with it.

In general, the internal representation will be temporally offset from the SoccerServer cycle by some amount, $\tau$, and communication between agent and SoccerServer will be delayed by some amount, $\Delta$. Therefore the optimal time to act relative to the internal clock is $t^* - \tau - \Delta$, where $t^*$ is the optimal external time to command action (immediately before the SoccerServer simulates the environment’s dynamics). The algorithm schedules action commands at this optimal internal time, which is estimated using separate estimates of $\tau$ and $t^*$, which are assumed to encompass the communication transit time:
5.1.1 Estimating the Internal Cycle Offset

The internal cycle offset is estimated in the Estimate Offset module by modelling noise in proprioception times. Although the SoccerServer specification states that proprioception occurs at the beginning of every cycle, actual perception times fluctuate due to varying network reliability and SoccerServer timer errors. As seen in Figure 5.3, these fluctuations can be quite severe, making it difficult to infer the SoccerServer cycle from perception times alone.

As opposed to external synchronization (which assumes no transmission delay) our model is based on the conservative assumption that a quick transmission is likely to occur within a small number of simulation cycles [Cri89, Bol93]. If we assume that at least one out of $N$ messages will be delivered promptly, then a good estimate of the cycle offset is given by Equation 5.3, where $t^{(i)}_{prop}$ is the internal perception time of the $i$th previous proprioception.

$$
\hat{\tau} = \min_{i=0}^{N-1} \left\{ t^{(i)}_{prop} \right\}
$$

Although this estimate is necessarily an over-estimate (because it includes
the transit time of the quickest message), the bias can be neglected by considering the necessary communication delays while estimating the optimal time to command action. Further, the estimator is robust to intermittent communication delays, as shown in Figure 5.3. Sustained delays are not characteristic of network noise [Bol93].

![Observed cycle periodicity](image)

Figure 5.3: Typical variation in proprioception times (in gray) recorded by measuring agent `sense_body event` frequency during a game. Also shown is the resulting estimate of the internal cycle offset, in bold.

### 5.1.2 Estimating the Optimal Time to Command Action

Because both network delays and processor strain vary over time, so too does the optimal time to command action. Therefore, it is beneficial to adapt the estimate of the optimal command time during operation. This is performed by the `Update t* belief` module.

Although the optimal time can not be directly observed, the validity of the current estimate can be inferred from action successes and failures. The `SoccerServer` effects action based on the external time of the action request (according to the rules in Table 2.1); this principle can be reversed to infer the external time from action successes and failures, providing a coarse validity observation of the current estimate, as is done in Table 5.1.
Table 5.1: Inferring the validity of the optimal command time estimate assuming known cycle offset. Because inference operates on the timescale of cycles (where communication delay can be deemed negligible) and does not depend on proprioception, its reliance upon network responsiveness is minimal.

<table>
<thead>
<tr>
<th>Action Confirmation</th>
<th>Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effected on time</td>
<td>$\hat{t}^* \leq t^*$</td>
</tr>
<tr>
<td>Effected late</td>
<td>$\hat{t}^* &gt; t^*$</td>
</tr>
<tr>
<td>Action failed</td>
<td>$\hat{t}^* \approx t^*$</td>
</tr>
</tbody>
</table>

Once the validity of the current estimate is inferred, the estimate can be modified accordingly. However, there are still two important issues:

1. The inference obtained from any individual action detection can be adversely affected by spurious communication delays. This side effect is avoided by maintaining a probabilistic distribution over inference conclusions, based on a stochastic model of spurious communication.

2. Once a conclusion is reached, how much the estimate should be changed is unknown. For example, if the estimate is less than optimal then it can be increased (thereby increasing time for sensing and deliberation) while still commanding action on time. However, because the optimal time is not observable, there is no immediate indication whether a particular increase moves toward the optimal or past it.

As seen in Figure 5.4, appropriate parameterization of these solutions can produce desirable results.

5.1.3 Confirming Commanded Action

The validity of the optimal command time estimate is inferred from the outcome of commanded action, which is determined in the Confirm Pending Actions module. When an action is commanded, the module adds it to a chronological list of pending action commands, $P$. When the agent can confirm that the action was effected or has failed, the command descriptor is moved to a list of confirmed action commands, $C$. If the agent assumes that action commands are received by the SoccerServer in
Figure 5.4: Adapting the estimate of the optimal time to act during operation using the parameters used by the UBCDynamo02 team during competition. The faint line represents the optimal time to command action, and the darker ones represent dynamic estimates with different prior values.

If the same order that they were sent,\(^1\) then confirmation follows from a specific causal relationship between time and the count of effected actions, \(n_e\), which is provided as part of the proprioceptive stimulus. The \(i^{th}\) pending action is confirmed if the count of effected actions is not less than the number of action commands preceding it:

\[
n_e \geq ||C|| + i \quad \rightarrow \quad P[i] \text{ confirmed}
\]  \hspace{1cm} (5.4)

Once an action command has been confirmed, its status (whether it was effected on time, late, or not at all) is determined. If a single cycle elapses before confirming a command, then that action was effected on time. However, if more than one cycle elapses, it can be difficult to distinguish between late and failed actions. The distinction can only be made by imposing a bound on communication delay, \(\Delta_{\text{max}}\), above which unaccounted action requests are assumed lost.

\(^1\)This assumption is not guaranteed by the UDP/IP transport layer, but it is likely since subsequent action commands are sent in approximately 100 millisecond intervals.
Table 5.2: Inferring the status of the $i^{th}$ pending action command from the count of effected actions.

<table>
<thead>
<tr>
<th>Effected action count</th>
<th>Cycles elapsed</th>
<th>Action status</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_e \geq i$</td>
<td>1</td>
<td>Effected on time</td>
</tr>
<tr>
<td>$n_e \geq i$</td>
<td>$[2, 1 + 2 \left[ \frac{\Delta_{\text{max}}}{100} \right]]$</td>
<td>Effected late</td>
</tr>
<tr>
<td>$n_e &lt; i$</td>
<td>$[1 + 2 \left[ \frac{\Delta_{\text{max}}}{100} \right], \infty)$</td>
<td>Failed</td>
</tr>
</tbody>
</table>

However, ambiguities arise if a new action is commanded before a previous action has been confirmed. This often occurs while waiting to distinguish between late and failed action and, in such circumstances, all hypotheses must be maintained until the ambiguity can be resolved or can be proved to be undecidable.\(^2\)

The basis of disambiguation are the strict rules that govern how an action command is handled by SoccerServer when it is received (see Table 2.1). In particular, the agent can exploit the fact that an action cannot be effected on time immediately after a late action to derive the disambiguation rules in Equations 5.5 to 5.7.

\[
P[i] \text{ eff} \text{ected on time} \rightarrow \forall j \leq i \ (P[j] \text{ eff} \text{ected on time}) \quad (5.5)
\]
\[
P[i] \text{ eff} \text{ected late} \lor n_e = \|C \cup P\| \rightarrow \forall j \geq i \ (P[j] \text{ eff} \text{ected late}) \quad (5.6)
\]
\[
i = \|P\| \rightarrow \forall j \leq i \ (P[j] \text{ undecidable}) \quad (5.7)
\]
\[
\|C \cup P\| \text{ of which are lost}
\]

5.2 Interpreting Observations

Within the agent architecture described in Chapter 3, the Filter module implements a Bayesian filter based on a particular, probabilistic model of the environment which, for the RSL agent, will be the intentional model introduced in Chapter 4. The model, combined with Filter, uses percepts as evidence to infer distributions over a collection of random variables (including intentional characteristics of observed

---

\(^2\)Ambiguities are only undecidable when a failed action cannot be assigned to a specific pending command. This does not matter for the purpose of $t^*$ estimation because failure is certain, but this does effect model prediction and activity analysis since the agent does not know which action failed.
agents) that constitute the observer's internal representation of its surroundings. The complete model is specified by a number of domain-dependent parameters:

1. The contents of the world state vector.
2. A prior probability distribution over the modelled random variables.
3. A stochastic model of perception.
4. A stochastic model of state transition (i.e., the environment dynamics).
5. The intentional extension of the model is domain-independent and fully specified in Chapter 4. However, there are situation-dependent parameters: the number of observed agents, individual intentions, and joint intentions considered.

The first four points of this specification are given for the RSL domain in the following sections starting with Section 5.2.1 which suggests initial distributions of world states. Given an initial description of the world state, the effect of time (in Section 5.2.2) and observations (in Section 5.2.3) on the world state are then specified.

The situation-dependent values, however, are parameterized using cross-validation for particular situations within this thesis. Future work may include having the agent manage these values online, for example.

### 5.2.1 Initial World States

The first step in modelling the system is to specify its state vector — a set of variables that are able to distinguish between important situations in the world — along with probable initial states. Consider, for example, the RSL domain world state given in Section 2.3. In the RSL domain, there are a number of constraints that can be used to construct a prior probability distribution over the initial values of this world state.

- The ball is exactly at the centre of the pitch.
- Opponents are on the opposite half of the pitch from the agent. Although it is not likely that initial positions would be distributed uniformly over this domain, it is safest to assume so given no information about the opponent team.
• Teammates are on the same half of the pitch as the agent. In this case, however, it is more likely to assume that the agent knows the initial configuration of the team.

• All objects are stationary at their initial locations.

5.2.2 World State Transitions

The next step in modelling the system is to specify its dynamics: how the world state vector changes through both action and inaction. In the RSL, all objects independently adhere to the same parameterized dynamic equations described in Section 2.4.1. These equations can be generalized into the stochastic state transition model below.

\[
\Pr(X^o_{t+1}, V^o_{t+1} | Y_{0:t}) = \int \ldots \int \Pr(X^o_t, V^o_t | Y_{0:t}) \Pr(X^o_{t+1} | X^o_t, V^o_t) \times \int_{F^o_{t+1}} \Pr(F^o_{t+1} | X^o_{t+1}, X^o_t, V^o_t, Y_{0:t}) \Pr(V^o_{t+1} | V^o_t, F^o_{t+1}) \quad (5.8)
\]

Equation 5.8 is only non-zero if the equations of dynamics (Equations 2.17 and 2.18) are obeyed. When they are, the transition can be described using Equation 5.9.

\[
\Pr(X^o_{t+1} = \bar{x}_{t+1}, V^o_{t+1} = \bar{v}_{t+1} | Y_{0:t}) = \int_{F^o_{t+1}} \Pr(X^o_t = \bar{x}_{t+1} - \frac{\bar{v}_{t+1} - \frac{1}{m} \bar{f}}{1 - \mu}, V^o_t = \frac{\bar{v}_{t+1} - \frac{1}{m} \bar{f}}{1 - \mu} | Y_{0:t}) \times \Pr(F^o_{t+1} = \bar{f} | X^o_{t+1} = \bar{x}_{t+1}, X^o_t = \bar{x}_{t+1} - \frac{\bar{v}_{t+1} - \frac{1}{m} \bar{f}}{1 - \mu}, V^o_t = \frac{\bar{v}_{t+1} - \frac{1}{m} \bar{f}}{1 - \mu}, Y_{0:t}) d\bar{f}
\]

(5.9)

However, specifying the impulse model, \(\Pr(F^o_{t+1} | X^o_{t+1}, X^o_t, V^o_t, Y_{0:t})\) is difficult as most forces are exogenous. Any model falls between two extreme strategies:

1. Optimistically consider any action that possibly affects the object. This strategy is guaranteed to maintain consistent beliefs, though confidence can quickly decay.

2. Pessimistically consider no action effect on the object. This strategy can result in inconsistent beliefs when the object is acted on so a consistency check is
needed to discard inconsistent hypotheses. However, when the assumption holds this strategy maintains higher confidence and accuracy.

The ideal model follows some intermediate strategy. For example, the agent should be more discriminatory about its own actions (which can be confirmed using the Schedule module) and develop beliefs about likely exogenous actions. Such a model of exogenous actions is developed in Chapter 4.

5.2.3 World State Perception

Another step in modelling the system is to specify the effect of observations on the believed distribution over world states. When an observation is made, it is considered using a stochastic model of perception based on Equations 2.19–2.21. Because the ceiling operator compresses information, the nature of this mapping is many to one: each point in percept space maps to a region in the world. If (as elaborated in Section 7.1) the ceiling operator is modelled as a stochastic function such that \( [a] - a \) is a random variable uniformly distributed in the range \([0, 1)\), then percept interpretation is a stochastic function from percepts into distributions over world states.

Specifically, the probability of an object being at a particular location \( \bar{x} \) after being observed with percept \( \hat{y}^z \) from location \( \bar{x}^a \) is only non-zero in the region defined by Equation 5.10, when it is defined by Equation 5.11.

If

\[
\frac{3}{20} < \frac{\|\hat{y}^z\|}{\|\bar{x} - \bar{x}^a\|} \leq \frac{3}{20} + \frac{\pi}{360} \quad \text{and} \quad \frac{\pi}{360} < \frac{\angle(\hat{y}^z)}{\angle(\bar{x} - \bar{x}^a)} \leq \frac{\pi}{20} + \frac{\pi}{360}
\]

then

\[
Pr(X^o = \bar{x}, X^a = \bar{x}^a, Y^o = \hat{y}) = \frac{1800 \sigma}{\pi} \ln \frac{\min\left\{ e^{\frac{3}{20}\|\hat{y}^z\|+1}, e^{\frac{3}{20}\|\hat{y}^z\|+1} \right\}}{\max\left\{ e^{-\frac{3}{20}\|\hat{y}^z\|+1}, e^{-\frac{3}{20}\|\hat{y}^z\|+1} \right\}}
\]

The above equation specifies the uncertainties involved with observing an object. Perhaps the most significant uncertainty is introduced by the observer’s own unknown position, since position and velocity observations are relative, uncertainty in its own state compounds the observer’s perceptual ambiguity. For this reason, self-localization becomes an important special case of perception, and will be described in the following section.

Finally, the critical issue of identification, which has been largely overlooked, is addressed in Section 5.2.3.2.
5.2.3.1 Self-localization

Self-localization is the procedure of determining the agent's own position, and is a critical factor in reconstructing beliefs from agent-centred observations. In the RSL, an agent localizes itself by observing landmarks (see Section 2.3.3). Although each landmark can be filtered individually, pose recovery using multiple landmarks is a characteristically different problem. Figure 5.5 shows the individual distribution bounds for each observed landmark for a particular agent pose, but each observation is not independent because they are all observations of a single object: the world relative to the agent (or equivalently, the agent relative to the world).

Figure 5.5: A collection of landmarks perceived in a single time step. Black disks represent each landmark's true location, \( \bar{x}_l \), squares represent each landmark's perceived location, \( \hat{x}_l \), and shaded regions represent non-zero portions of the posteriors, \( \Pr(X|Y, X^a) \).

Probabilistic self-localization from multiple landmarks is the determination of the posterior distribution over agent poses, given the observations of a number of landmarks, and their known location. For \( L \) landmarks,

\[
\Pr(X^a|Y^L, X^L) = \Pr(X^a|X^L) \prod_{k=1}^{L} \frac{\Pr(Y^L_k|X^a, Y^L_{k+1}, X^L_{k+1})}{\Pr(Y^L_k|Y^L_{k+1})} \tag{5.12}
\]

Given the specific model of perception for each landmark (Equation 5.11), the agent can rely on the fact that each landmark percept only depends on the relative
position of the landmark with respect to the agent. In other words, $Y_{x^k} \perp X_{x^k} | X^a, X^l_i$ for every landmark $l_j$ other than $l_i$. This independence simplifies localization:

$$
Pr(X^a | Y_{x^k}^{1:L}, X^{l_1:L}) = Pr(X^a | X^{l_1:L}) \prod_{k=1}^{L} \frac{Pr(Y_{x^k}^l | X^a, X^l_i)}{Pr(Y_{x^k} | Y_{x^k}^{1:L-1})}
$$

(5.13)

Essentially, the probability of the agent being that location $x^a$ is equal to the probability of observing each landmark at its true location from $x^a$. This is computed for all prior possibilities to recover the distribution of agent positions. Although localization from a single landmark can be fairly uninformative, utilization of multiple observations in this manner makes the agent significantly more confident in its beliefs, as shown in Figure 5.6.

![Figure 5.6: An example of using multiple observations to reduce perception ambiguity. The polygons represent the extent of a probabilistic distribution over possibilities obtained from a single percept (the distribution's value is known, Equation 5.11, but not shown). Considering all landmarks, only a small portion of the domain remains with non-zero probability.](image)

5.2.3.2 Object identification

Often, an observation cannot be attributed to any single object unequivocally. In the RSL domain, for example, identification reliability degrades with distance from the observer. However, correct association of percepts to particular objects is necessary for accurate modelling in this system. Under ambiguity, hypothesis
postulation is based on the noise model governing perception. For the RSL, there are three relevant features of perception:

1. each object can be observed at most once per cycle,
2. each observation is of one object, and
3. an observation constrains the relative location of the object in accordance with Equation 5.10.

These features can be exploited, engendering the rules for generating hypotheses listed in Table 5.3 and used in the example in Figure 5.7. As more observations are made, the probability of each hypothesis is re-evaluated and hypotheses that become sufficiently improbable are pruned. In practice, very few ambiguities are perpetuated for long periods of time.

Table 5.3: Rules used for labelling observations when identification is ambiguous.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int \Pr(\mathbf{X}^O = \tilde{x}</td>
<td>\mathbf{X}^a, \mathbf{Y}^{z^O} = \tilde{y}) \Pr(\mathbf{X}^O = \tilde{x}) , dx = 0$</td>
</tr>
<tr>
<td>else if $\forall Q \neq O \left( \int \Pr(\mathbf{X}^Q = \tilde{x}</td>
<td>\mathbf{X}^a, \mathbf{Y}^{z^Q} = \tilde{y}) \Pr(\mathbf{X}^Q = \tilde{x}) , dx = 0 \right)$</td>
</tr>
<tr>
<td>else</td>
<td>Consider object $O$ as a possible match with observation $\tilde{y}$</td>
</tr>
</tbody>
</table>

5.3 Deliberation

The highest level of deliberation is planning: the consideration, evocation, and revision of the observer’s own intentions. Based on the intentional stance view of agency, this process can be modelled as a sequential decision process from believed random variables to intentions. Specifically, the observer’s policy maps specific states of its internal representation directly into an intention. The policy can address each element of the internal representation’s cross product separately, or map larger classes into single intentions. Several intentions were designed for the RSL domain and are available for the agent to choose from, as listed below. However, a rather
simple policy was used (see Figure 5.8) that only makes use of the world state vector. Though intentional characteristics are being inferred, their use in the observer’s policy is left for future work.

**NoIntention** The agent has no commitment to pursue any particular goal.

**ScanForInformation** The agent commits to gathering information about the world. The commitment is revised when information of interest is acquired, or it is determined impossible to acquire that information (for example, by a change in the status of the game).

**KickInBall** The agent commits to performing the free kick awarded after an opponent foul. The intention is abandoned after the period allowed for a free kick has elapsed.

**KickInBallSupport** The agent commits to supporting the player who is performing a free kick. The intention is abandoned when the free kick is completed.

**RunOnToBall** The agent commits to intercepting the ball. The intention is abandoned when the agent no longer believes that it can intercept the ball.

In general, each intention corresponds to a list of constraint conjunctions that collectively specify the intent. For example, the sections from 5.3.1 to 5.3.11 describe a number of constraints that were tested and used to compose general soccer behaviour. They are summarized in Table 5.5. Therefore, intention satisfaction takes the form of satisfying constraints.

The agent’s constraint solver is implemented by the Means-ends reasoning module, which performs gradient ascent in the space of actions (described in Table 5.3) to find the action that would most likely achieve a world state that satisfies
Figure 5.8: An example of the decision process used by the goalie for the UBC Dynamo02 team in competition [Mon02]. In this case, the agent's beliefs were mapped to logical predicates and the decision process can be represented as a decision tree. Solid lines and dashed lines indicate truth and false conclusions respectively.

In summary, deliberation within the Planner module commits to an intention which is described by a list of constraint conjunctions. Means-ends reasoning then produces a sequence of actions that moves the system into a solution state, satisfying the intention.

5.3.1 The Scan Constraint

The Scan constraint is designed to maximize the amount of information obtained from the agent's current position. In this case, new information is defined as elements of the domain that the agent cannot see; the amount of new information obtained through action is calculated using Equation 5.14.
Table 5.4: Description of RSL actions used by the Means-ends reasoning module to approach constraint solutions. $p_{\text{max}}, m_{\text{max}},$ and $k_{\text{max}}$ are all SoccerServer parameters indicating maximum agent power, moment, and kicking power respectively [CFH+02].

<table>
<thead>
<tr>
<th>Action</th>
<th>Expected result</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>dash $p$</td>
<td>$\ddot{x}_t^a = p \leq \theta_t^a$</td>
<td>$p \in [0, p_{\text{max}}]$</td>
</tr>
<tr>
<td>turn $m$</td>
<td>$\theta_{t+1}^a = \theta_t^a + \frac{m}{1+|\dot{v}_t^a|}$</td>
<td>$m \in [-m_{\text{max}}, m_{\text{max}}]$</td>
</tr>
<tr>
<td>turn.head $m$</td>
<td>$\phi_{t+1}^a = \phi_t^a + m$</td>
<td>$m \in [-m_{\text{max}}^n, m_{\text{max}}^n]$</td>
</tr>
<tr>
<td>kick $\ddot{p}$</td>
<td>$\ddot{x}_t^a \propto \ddot{p}$</td>
<td>$|\ddot{p}| \in [0, p_{\text{max}}] \land</td>
</tr>
</tbody>
</table>

Table 5.5: Summary of constraints. Refer to the noted section for more detail.

<table>
<thead>
<tr>
<th>Constraint Name</th>
<th>Section</th>
<th>Relations to satisfy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan</td>
<td>5.3.1</td>
<td>Maximize new information with constant $\ddot{x}^a$.</td>
</tr>
<tr>
<td>LookAround</td>
<td>5.3.2</td>
<td>Maximize new information with constant $\ddot{x}^a, \theta^a$.</td>
</tr>
<tr>
<td>LookAt</td>
<td>5.3.3</td>
<td>Maximize new information with constant $\ddot{x}^a, \theta^a$ and object $O$ within the field of view.</td>
</tr>
<tr>
<td>BeforeKickoff</td>
<td>5.3.4</td>
<td>Maximize new information with constant $\ddot{x}^a, \theta^a$ and $\ddot{x}^a$ equal to the player’s initial position.</td>
</tr>
<tr>
<td>GoTo</td>
<td>5.3.5</td>
<td>$\ddot{x}^a = \text{desired location}$.</td>
</tr>
<tr>
<td>InterceptBall</td>
<td>5.3.6</td>
<td>$|\ddot{x}^a - \ddot{x}^b| \leq \text{kickable distance}$.</td>
</tr>
<tr>
<td>PlayerPosition</td>
<td>5.3.7</td>
<td>Attain free kick position, then acquire maximum information (prioritizing ball information).</td>
</tr>
<tr>
<td>GoaliePosition</td>
<td>5.3.8</td>
<td>$T_d^b T_o^g = T_o^g T_b^g$, see Section 5.3.8.</td>
</tr>
<tr>
<td>DefensivePosition</td>
<td>5.3.9</td>
<td>$\ddot{x}<em>t^a = \ddot{x}</em>{t-1}^a + w_1 \ddot{a}_1 + w_2 \ddot{a}_2$</td>
</tr>
<tr>
<td>MiddlePosition</td>
<td>5.3.10</td>
<td>Mark or evade in the middle of the field.</td>
</tr>
<tr>
<td>OffensivePosition</td>
<td>5.3.11</td>
<td>Mark or evade at the team’s offside line.</td>
</tr>
</tbody>
</table>
The Means-ends reasoning module attempts to maximize this information by rotating its head or body (modifying $\theta^a$ and $\phi^a$ respectively). However, because the sensor-limited agent cannot view the entire environment, the constraint will never be solved. It is useful, nonetheless, because the Means-ends reasoning module will produce a sequence of useful actions in an attempt to solve it.

5.3.2 The LookAround Constraint

The LookAround constraint is designed to maximize the amount of information obtained from the agent's current position and heading. Specifically, it is the conjunction of the Scan constraint (Section 5.3.1) with Equation 5.15, the latter of which has priority.

$$\theta^a_{t+1} = \theta^a_t$$

(5.15)

5.3.3 The LookAt Constraint

The LookAt constraint is designed to maximize the amount of information obtained from the agent's current position and heading, while maintaining a particular object, $O$, within its field of view. Specifically, it is the conjunction of the LookAround constraint (Section 5.3.2) with Equation 5.16, the latter of which has priority.

$$\angle(\bar{x}^0 - \bar{x}^a) - \theta^a - \phi^a \in \text{field of view}$$

(5.16)

5.3.4 The BeforeKickoff Constraint

The BeforeKickoff constraint is designed to position the agent and initialize its memory. The constraint combines Scan (Section 5.3.1) to maximize the amount of information in the initial memory with Equation 5.17 to attain the player's initial position on the pitch, $\bar{x}_0^a$. Positioning is prioritized over information retrieval.

$$\bar{x}^a = \bar{x}_0^a$$

(5.17)
5.3.5 The GoTo Constraint

The GoTo constraint is designed to move the agent to a desired position. The Constraint Optimization module attempts to achieve this constraint by applying appropriate forces to the agent’s body in accordance with the equations of dynamics (Equations 2.17 and 2.18). It can calculate the force required to move the agent to the desired position, but there are a number of restrictions upon what force the agent can actually effect, all of which are crucial to optimal motion design.

1. There is a maximum bound on the force magnitude, which depends on the agent’s current stamina level.
2. There is a maximum bound on the velocity that the agent is able to achieve.
3. The agent can only accelerate in the direction of its current heading.
4. The agent has a limited reserve of stamina, which is depleted by each applied force, and replenished through inaction.

If the desired force is initially insurmountable then its effect must be applied over time. For example, if the third constraint is violated, the agent must turn itself to the desired direction before accelerating. Since turning takes time, the agent must project its position into the future to determine the force that will take the agent to the desired position. This, of course, presents further complications, including a limit on the amount the agent can turn depending on its velocity, as well as sensor and actuator noise that may inhibit the agent from effecting actions exactly as desired.

To decompose the desired force over an extended operation, the agent must consider its momentum and project its position and velocity into the future. Based on a constant acceleration strategy, this projection is performed using Equations 5.18, and 5.19 which refer to all mobile objects.

\[
\begin{align*}
\ddot{x}_{t+T} &= \ddot{x}_t + \sum_{i=0}^{T-1} (1 - \mu)^i \left( \ddot{v}_t + \frac{T-i-1}{m} \ddot{f} \right) \\
\ddot{v}_{t+T} &= (1 - \mu)^T \left( \ddot{v}_t - \frac{1}{m\mu} \ddot{f} \right) + \frac{1}{m\mu} \ddot{f}
\end{align*}
\]  

Unfortunately, there is no analytic solution for both the length of the trajectory, \(T\), and desired applied force, \(\ddot{f}\). The agent must heuristically search the trajectory space for solutions, seeking the quickest overall trajectory that yields a desired force that is valid with respect to the maximum acceleration, maximum speed, and available stamina. For example, see Figure 5.9.
5.3.6 **The InterceptBall Constraint**

The *InterceptBall* constraint is designed to move the agent to within a kickable distance of the ball. The constraint is satisfied similarly to the *GoTo* constraint where the desired position is the ball’s location when the trajectory completes. The ball’s position can be projected into the future using Equation 5.18 (assuming no exogenous actions) and the agent can choose the quickest trajectory satisfying Equation 5.20 to optimally intercept the ball.

\[
\| \tilde{x}_t^a - \tilde{x}_t^b + \sum_{i=0}^{T-1} (1 - \mu)^i \left( \tilde{v}_t^a - \tilde{v}_t^b + \frac{T-i-1}{m} \tilde{f}_a \right) \| \leq \text{kickable distance} \quad (5.20)
\]

The assumption that no force is applied to the ball is important, and it is critical for an agent to judge how likely this is. The agent can calculate an lower bound on the amount of time available to intercept the ball by considering how long it will take each other player to intercept the ball optimally\(^3\). An agent should only invoke the *InterceptBall* constraint if it is likely that it will intercept the ball.

5.3.7 **The PlayerPosition Constraint**

The *PlayerPosition* constraint is designed to produce fundamental soccer-player behaviours; tasks that a player in any role might perform. These responsibilities are

\(^3\)Players can accelerate with a *SoccerServer*-specified maximum acceleration for a finite period of time that is governed by stamina. When a player’s stamina is sufficiently depleted, its maximum acceleration is dramatically reduced and its ability to replenish stamina is permanently restricted. For these reasons, very few teams allow their agent’s to accelerate beyond this critical stamina threshold. Although it would be beneficial to model and infer other agent’s stamina, the UBCDynamo02 agent conservatively assumes that every agent has full stamina.
specifically in belief maintenance and positioning during certain set plays (for example, during a free kick). More complicated roles can then be constructed that make use of this default behaviour (see Sections 5.3.8 to 5.3.11). \( PlayerPosition \) is the concatenation of the following constraints, in order of their priority:

1. \( GoTo \), with the following desired position:
   - If the agent is performing a kick-in, the desired position is directly behind the ball (facing the pitch) and within kickable distance from the ball.
   - If the agent is performing a free kick, the desired position is directly behind the ball (facing the opponent’s goal) and within kickable distance from the ball.
   - If the agent is supporting a kick-in or free kick, the desired position is the nearest position at which the passed ball will likely be received. This is determined by knowing how the teammate will kick the ball, and searching for an appropriate initial position based on Equation 5.20.

2. Equation 5.16 with \( O = \text{ball} \). In other words, to maintain accurate beliefs in the state of the world (maximizing viewable area over time) with a priority on ball information (by keeping the ball within agent’s field of view) from the agent’s current position.

### 5.3.8 The GoaliePosition Constraint

The \( \text{GoaliePosition} \) constraint is designed to place a player in a position capable of preventing the ball from entering its own goal. The basic strategy is to only move parallel to the goal line at a particular distance away from its goal. \( \text{GoaliePosition} \) is the concatenation of the following constraints, in order of their priority:

1. \( \text{PlayerPosition} \)

2. To face the ball:

   \[
   \phi^a = \angle(\bar{x}^b - \bar{x}^a) - \theta^a
   \]  
   \[ (5.21) \]

3. \( GoTo \), where the agent’s desired position is such that it can reach a point on any, scorable ball trajectory before the ball does. To reduce the search space, the agent only considers ball interception at the closest trajectory location to its desired location. Under this restriction, interception of scorable ball trajectories is guaranteed by the constraint in Equation 5.22, where \( T_x^0 \) is the
time it will take object $o$ to reach point $\tilde{x}$, and the points $\tilde{l}$ and $\tilde{r}$ are the two maximal interception points as illustrated in Figure 5.10.

$$T_l^b \geq T_l^a \land T_r^a \geq T_r^a$$

(5.22)

Figure 5.10: Illustration of the desired position used by the GoaliePosition constraint.

In the worst case, the ball may be kicked from its current position with its maximum possible speed, $\|\tilde{v}_b\|_{\text{max}}$, along the left-most or right-most trajectory. The agent assumes that it will accelerate with maximum force toward the appropriate interception location, and that it will not be limited by its maximum velocity or lack of stamina. This assumption is likely because the goalie is relatively inaccurate and the interception point is relatively close (on the order of the width of the goal). Given these situations, Equation 5.18 can be used to determine the duration of the ball’s and the agent’s trajectory:

$$T_x^\phi = \frac{\ln \left\{ 1 - \frac{\mu}{\|\tilde{v}_b\|_{\text{max}}} \|\tilde{x} - \tilde{x}_b\| \right\}}{\ln \{1 - \mu\}}$$

(5.23)

$$\mu\|\tilde{x} - \tilde{x}_a\| = (1 - (1 - \mu)^{T_x^\phi}) \|\tilde{v}_a\| + \left[ T_x^a - 1 - \frac{1 - \mu}{\mu} (1 + (1 - \mu)^{T_x^a - 2}) \right] a^a_{\text{max}}$$

(5.24)

Unfortunately, the constraint cannot be solved with a closed-form equation: the agent must search the space of possible desired locations. The agent heuristically searches along one of the maximal ball trajectories. Given an interception point along this trajectory, the agent’s desired location is constrained to
the point's — only a small segment of which will satisfy both the first and second conjugates of the constraint. In general, there may be a range of solutions for any interception point. The agent prefers the mean of the satisfying set to alleviate errors in perception and those caused by the invalidity of the above assumptions.

5.3.9 The **DefensivePosition** Constraint

The *DefensivePosition* constraint is designed to hamper a player's progress toward the agent's goal. This is done by physically impeding the player's trajectory and by maintaining aggressive offside positioning. *DefensivePosition* is the concatenation of the following constraints, in order of their priority:

1. **PlayerPosition**

2. **GoTo** where the agent’s desired position is the weighted vector sum of two attractions \( \vec{a} = \vec{x}_a + w_1 \vec{a}_1 + w_2 \vec{a}_2 \). The first attraction is toward the ball.

   \[
   \vec{a}_1 = \vec{x}_{t+1} - \vec{x}_t \tag{5.25}
   \]

   The second attraction depends on the agent's **mark** (the closest opponent with this agent being the closest teammate). If the mark is within some threshold of the agent (e.g., 10 metres) then the second attraction is 2 metres away from the mark, between the opponent and the ball. Otherwise, the second attraction is to one of a pencil of lines emanating from a point 20 meters directly behind the agent’s own goal, parameterized by the agent’s defensive role \( \gamma \).

   \[
   \vec{a}_2 = \begin{cases} 
   2\vec{x}_b - \vec{x}_p & \text{if player } p \text{ is the agent's mark,} \\
   \begin{bmatrix}
   0 & 0 \\
   \gamma & -1 
   \end{bmatrix} \vec{x}_a + \begin{bmatrix} 0 \\ 20\gamma 
   \end{bmatrix} & \text{else.} 
   \end{cases} \tag{5.26}
   \]

5.3.10 The **MiddlePosition** Constraint

The *MiddlePosition* constraint is designed to facilitate the movement of the ball through the middle of the pitch toward the opponents goal, and to disrupt its movement toward the agent’s goal. *MiddlePosition* is the concatenation of the following constraints, in order of their priority:
1. PlayerPosition

2. GoTo where the agent’s desired position depends on which team has possession of the ball.

- If the opponent team has possession of the ball, the agent’s desired position is the marking position of its mark (or, if it has no mark, of the closest unmarked opponent).
- A middle player, by definition, plays in the area between the teammate closest and furthest from its own goal line. The role of a middle player is parameterized with an off-centre distance, $y$, and a breadth percentage, $x$. If the agent’s team has possession of the ball, then these parameters define its base desired position as $y$ meters tangent to its goal line, and $x$ percent across this allowed area (see Figure 5.12). Appended to this base desired position are a set of repulsive forces, weighted inversely by the opponents distance from the agent, that push the agent away from each opponent, but only tangent to its goal line.

5.3.11 The OffensivePosition Constraint

The OffensivePosition constraint is designed to move the agent at the most opportune location to receive the ball and move it to the goal. OffensivePosition is similar to MiddlePosition except that its base desired position places it on the team’s offside
Figure 5.12: Illustration of the constraint on an agent's position provided by *MiddlePosition* when its teammates have possession of the ball (left) and when the opponents have possession of the ball (right).

line\(^4\) as shown in Figure 5.13.

Figure 5.13: Illustration of the constraint on an agent's position provided by *OffensivePosition* when its teammates have possession of the ball.

### 5.4 The RoboCup Simulation League Agent

The agent described in this chapter was implemented using C++, and tested on a Linux-based computer with *SoccerServer* version 8.5. Although specific elements of the agent are tested in the next chapter, the agent in general was able to produce rational soccer behaviours and competed respectably, placing 26\(^{th}\) with 42 teams from around the world in the 2002 Robot World Cup competition.

\(^4\)The offside line is a line parallel to the goal line, that intersects the opponent player or ball that is furthest up-field.
Chapter 6

Experimental Results For The RSL Agent

This chapter presents a series of controlled experiments designed to test the reactivity of the RSL agent described in Chapter 5, along with its ability to infer an accurate internal representation of its environment. These capabilities are fundamentally important to an agent and are each shown to be proficiently performed through the results of the orthogonal experiments.

The first experiment, which is discussed in Section 6.1, was performed to test the agent's reactivity by exploring the adaptive deliberation scheduling algorithm described in Section 5.1. The ability to thus synchronize with the environment is crucial as it forms the foundation upon which the rest of the architecture is built. Indeed, synchronization during realistic scenarios is validation of the sufficient reactivity provided by the architecture as a whole.

The remaining experiments were performed to test the agent's ability to infer an appropriate internal representation. First tested, as discussed in Section 6.2, is the lowest hidden layer of the intentional model (i.e., the position and velocity of all 23 objects). The accuracy of these inferences necessary as the observer's control is known to rely heavily its believed world state. Furthermore, as discussed in Section 6.2.4, experiments were performed to test the agent's ability to infer high-level characteristics about its environment such as the intentions of other agents (using the full model of Chapter 4). Although this thesis only assumes that the ability to infer abstract behavioural descriptions can play an important role in the agent's higher-level planning and strategy adoption, the results qualitatively suggest that appropriate high-level characteristics can be inferred from the environment. It is left for future work to fully incorporate these inferences into the agent's decision process and obtain quantitative confirmation of the assumption.
6.1 Adaptive Resource Scheduling

To test the agent's reactivity, the SYNCHRONIZATION experiment involves teams of agents (each using a different synchronization algorithm) competing in a soccer game. During the game, the inferred status\(^1\) (whether an action was effected late, on time, or not at all) is recorded for each action requested. Over several games, this data begins to reveal important characteristics of the algorithm, shown in Figure 6.1.

![Observed Action Correctness](image)

Figure 6.1: Distributions of correct, late, and failed actions observed by agents using the internal, external, and dynamic action index synchronization algorithms during the SYNCHRONIZATION experiment. Each synchronization algorithm was tested using a network that introduced varying transmission delays averaging approximately 15 milliseconds.

The high variance observed using internal synchronization is significant. It corresponds to the uniform distribution of possible cycle offsets: when the agent starts its cycle is not correlated at all with the SoccerServer's cycle. If the internal cycle happens to be synchronized with the external cycle the agent is able to perform a high fraction of correct action. However, poor synchronization is just as likely.

External synchronization was observed to produce fewer correct actions than the other algorithms on average. For external synchronization, variance from the average is a result of sensitivity to prompt proprioception: small changes in commu-

\(^1\)The online procedure described in Section 5.1.3 automatically provides expressive information concerning the agent's synchronization - an ability which is a special feature of the online algorithm that combines the intended action time with its inferred status.
nication delay can cause a large change in the fraction of actions that are correct. Also, in a general setting it is unlikely that an external percept will actually signal the appropriate action time.

The dynamic action index synchronization algorithm produces the most correct actions on average. Also, the observed variance is low because the evolving estimates continually converge on optimal values making the algorithm robust to varying conditions.

6.2 Interpreting Observations

This section describes experiments that test the agent’s ability to infer the value of a collection of random variables (described by the model in Chapter 4) based on observations of its environment. During this process, the agent evolves a filtering distribution based on observations as they arrive. The purposes of these experiments are to determine the accuracy of the model developed in this thesis by comparing this distribution to ground truth (where available) and expected results.

The accuracy of the random variables will be determined in a bottom-up order. The first two experiments test the agent’s ability to localize itself from a single visual percept (a single-observation kidnapped robot problem described in Section 6.2.1) and also during navigation (described in Section 6.2.2). These abilities are important since other world state dimensions are perceived through agent-centric sensors. The next experiment, presented in Section 6.2.3, tests the agent’s ability to perceive other objects moving independently throughout the environment. By tracking each of the 23 mobile objects in the environment, the agent can reconstruct the full world state vector.

The final set of experiments (Section 6.2.4) test the inference of the observed agents’ intentional attributes in a specific situation. This constitutes the whole of the observer’s internal representation.

6.2.1 Static Self–Localization

De Boer et al. have published results concerning the accuracy of self-localization in the RSL domain where they compare errors in agent’s believed position after localizing from a single observation [dBK02]. To compare with these results, their experiments are reproduced: the STATIC SELF–LOCALIZATION experiment involves repeatedly placing the agent in a random pose and allowing it to localize from one visual observation. The agent uses the same observation to localize using each of two filters:
1. An exact Bayesian filter using the "Two-landmark Uniform" distribution described in Section 2.3.3. Note that this is not identical to the approximate filter used in [dBK02], but should produce better results.

2. A sampled, approximate Bayesian filter using the distribution based on a probabilistic interpretation of the ceiling operator, described in Section 5.2.3.1.

After incorporating the observation, the mean of each distribution is compared with the agent's true pose to determine an absolute error. The results of several repetitions of this experiment are shown in Figure 6.2 and Table 6.1.

![Figure 6.2: Localization errors obtained from multiple iterations of the STATIC SELF-LOCALIZATION experiment.](image)

The first filter produces significantly better results than the two-landmark estimate alone by making use of the known bounds on possible locations to focus the two-landmark estimate onto more probable locations. The second filter is similar, except that it considers all possible locations, and does not treat these locations uniformly but weights them according to the perception model in Equation 5.11. This produces notably better results, suggesting that the probabilistic interpretation of the ceiling operator produces a distribution that is a more accurate model of perception in the RSL.
Table 6.1: Comparison of errors obtained from different filters used in the Static Self-Localization experiment.

<table>
<thead>
<tr>
<th>Filter type</th>
<th>Mean</th>
<th>Std. dev.</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-landmark uniform</td>
<td>9.45 cm</td>
<td>9.69 cm</td>
<td>0.320 mm</td>
<td>2.11 m</td>
</tr>
<tr>
<td>Probabilistic ceil.</td>
<td>5.91 cm</td>
<td>5.26 cm</td>
<td>0.222 mm</td>
<td>1.74 m</td>
</tr>
</tbody>
</table>

However, there is an important difference between these two filters: the first is exact; the second is approximate. The approximate filter involves more computation than the exact, uniform filter, and the computation time grows linearly with the number of delta functions used to approximate the distribution. It is important to consider how many delta functions are necessary to accurately approximate the distribution. During the Static Self-Localization experiment, the mean of the approximate filter's distribution was recorded based on different number of delta functions (multiples of 100), and the error in the resulting estimate recorded. The results are shown in Figure 6.3.

The accuracy of the distribution mean improves steadily as more delta functions are used. Results, of course, will vary depending on the random placement of those delta functions. Although this trend is clear, it is important to note that the difference in mean error when using 100 delta functions is less than a millimetre worse, on average, than using 5000 delta functions — and still better than the "Two-landmark Uniform" filter.

6.2.2 Dynamic Self-Localization

The results in Section 6.2.1 convey the benefit of the self-localization sensor described in Section 5.2.3.1. However, much of the power of Bayesian filtering comes in dynamic situations, where knowledge about the environment dynamics can be combined with observations over time. The Dynamic Self-Localization experiment involves repeatedly generating a random pose and allowing the agent to achieve that pose using the GoTo constraint (described in Section 5.3.5). The agent localizes itself using both internal beliefs as well as visual observations obtained during its trajectory, filtering five of the state's dimensions (corresponding to the agent's position, orientation, and velocity) using the second Bayesian filter. The mean of the resulting beliefs are compared with the true world state, resulting in
Figure 6.3: Expected localization errors caused by poor approximation of the true perception distribution. Note that the entire error range is less than one centimetre.

The errors displayed in Figure 6.4 and in Table 6.2. The dramatic improvements illustrate the benefit of filtering observations: the environment dynamics provide stringent constraints that restrict possibilities.

Table 6.2: Comparison of errors obtained from observations alone and from filtered observations during the Dynamic Self--Localization experiment.

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Mean</th>
<th>Std. dev.</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct position observation</td>
<td>6.11 cm</td>
<td>4.03 cm</td>
<td>1.00 mm</td>
<td>32.3 cm</td>
</tr>
<tr>
<td>Filtered position observations</td>
<td>3.48 cm</td>
<td>2.93 cm</td>
<td>0.295 mm</td>
<td>28.1 cm</td>
</tr>
<tr>
<td>Direct velocity observation</td>
<td>3.22 cm/s</td>
<td>1.74 cm/s</td>
<td>0.250 mm/s</td>
<td>11.8 cm/s</td>
</tr>
<tr>
<td>Filtered velocity observations</td>
<td>1.92 cm/s</td>
<td>2.67 cm/s</td>
<td>0.200 mm/s</td>
<td>16.6 cm/s</td>
</tr>
</tbody>
</table>
6.2.3 Object Tracking

The state also contains the position and velocity of the ball and 21 other players, each obeying the same parameterized dynamics and sensor models. The OBJECT TRACKING experiment instantiates an object with a random state (within the agent’s field of view), and allows the SoccerServer to effect the object until it comes to rest. During this interval, the agent remains stationary and filters its observations. When the object stops, a random impulse is applied to it and the process repeats. Throughout the experiment, the agent’s beliefs are compared to the true object state in Figure 6.5 and Table 6.3.

Table 6.3: Comparison of errors obtained from observations alone and from filtered observations during the OBJECT TRACKING experiment.

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Mean</th>
<th>Std. dev.</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct position observation</td>
<td>41.2 cm</td>
<td>31.9 cm</td>
<td>2.20 mm</td>
<td>2.31 m</td>
</tr>
<tr>
<td>Filtered position observations</td>
<td>30.6 cm</td>
<td>30.8 cm</td>
<td>0.40 mm</td>
<td>2.88 m</td>
</tr>
<tr>
<td>Direct velocity observation</td>
<td>56.8 cm/s</td>
<td>1.23 m/s</td>
<td>0.30 mm/s</td>
<td>62.5 m/s</td>
</tr>
<tr>
<td>Filtered velocity observations</td>
<td>4.66 cm/s</td>
<td>17.9 cm/s</td>
<td>0.00 mm/s</td>
<td>8.94 m/s</td>
</tr>
</tbody>
</table>
Figure 6.5: Results of the OBJECT TRACKING experiment showing errors in believed object position (left) and velocity (right) separately. The belief mean (solid) is compared to the percepts (dashed) showing the benefit of filtering observations.

Although the filtered position belief is more likely to produce low errors, the significant improvement observed in other dimensions of the world state is not observed. This is because simulation is sufficiently noisy that it contributes little information to the posterior, compared to the percept itself. An object’s velocity, however, is subject to less stochastic dynamics and the filter is able to tightly constrain the possibilities.

6.2.4 Modelling Multiple Agents

This section describes experiments involving the agent’s ability to filter its observations with respect to the full probabilistic model described in Chapter 4. The experiments’ domain is a subset of robot soccer: the M-ON-N ATTACK. During this scenario, $M$ teammates, who have possession of the ball, attempt to score on their opponent’s goal, which is defended by $N$ opponents and the opponent goalie. In a sequence, each player and the ball are placed randomly on the pitch and allowed to act until the ball leaves the pitch, scores, or is intercepted by a defender, or when a foul is committed. Other restricted domains have been proposed for the purposes of machine learning [SS02], but this scenario represents a more valid deconstruction of general soccer play, with natural means of scaling the problem [SAB+02].

The analysis begins by considering a model designed by a human with an intuitive understanding of the domain. This model, called Model-HC, was designed based on a small base of rules that describe typical 2-ON-0 ATTACK behaviour, in the eyes of the designer:
1. All group behaviours can be classified using two joint intentions:

   **intend_to_pass** where the ball carrier draws its opponent with the ball before passing to an open teammate.

   **intend_to_shoot** where the ball carrier singularly attacks the goal.

   These constitute the group's joint intentions formed arbitrarily but where **intend_to_pass** is considered four times more likely than **intend_to_shoot**.

2. All individual behaviours can be classified using three individual intentions:

   **intend_to_intercept** where the ball carrier tries to intercept the ball. A player with this intention is expected to only move and turn, and to do so until the ball has been intercepted.

   **intend_to_draw** where the ball carrier lures its opponent away from a teammate or the goal. A player with this intention is expected to dribble the ball until a good pass presents itself, at which time a pass is made.

   **intend_to_score** where the ball carrier singularly attacks the goal. A player with this intention is expected to dribble toward the goal until a viable shot presents itself, at which time a shot is made.

This model is fairly simple and largely deterministic. Nonetheless, it can characterize the 2-ON-0 ATTACK scenario with some level of accuracy.

### 6.2.4.1 Optimizing the Model Parameters

Despite the intuition that the model designer may hold, several factors detract from the hand-crafted model. For example,

- there are often variables that are actually hidden, i.e., their values or dynamics are not known and cannot be estimated;

- human memory and predictions are often unnecessarily general, leading to inaccurately estimated probabilities; and

- the space of behaviours is too complex to accurately characterize in a reasonable amount of time.

For these reasons, although educated guesses may suffice, it is often beneficial to incorporate observed experiences in a more formal way. For example, the 2-ON-0 ATTACK scenario was performed 900 times, producing 900 observation traces of
different lengths. This experience can be used to estimate optimal parameters for the probabilistic model, as described in Section 4.6.

To test this procedure, different subsets of the evidence are used to estimate the parameters of the model. The sequences left out of this training phase are then used to evaluate the learned model. This is repeated numerous times with different training and test sets. The purpose of performing these numerous experiments is to obtain a more accurate characterization of the learning algorithm and the learnable models, by effectively increasing the size of the data set. The steps of a single experiment are as follows:

1. Randomly divide the full data set of 900 sequences into mutually exclusive training and a test sets. The test set in this case was selected to be 10% of the size of the training set.

2. The sequences in the training set are then used to estimate model parameters that most likely describe the observed behaviour using the EM algorithm described in Section 4.6. For example, Figure 6.6 shows how Model-HC converges to a locally optimal parameter set (named Model-HC*) that demonstrates a 62440 magnitude improvement in likelihood.

3. The sequences in the omitted test set are then used to compare inferences based on the learned model. This validation step verifies the learned model's robustness, because the test sequence instances are not present in the training data.

Initial conditions can have a tremendous effect on the parameters that are estimated by this technique. In order not to limit the model testing to the region of parameter space closest (in a hill-climbing sense) to singular initial conditions, a random search through the parameter space was conducted. In this experiment, initial model parameters are selected at random (uniformly from the model space) and fit to the data set using the above procedure. Although the space of all model parameters is extremely large, such a search can be an effective strategy for finding a more successful local extrema.

The experiment was run with 113 random restarts. The local optimal parameter set from each of those experiments is displayed in Figure 6.7. The model with the highest likelihood — named Model-Best — is a 411 magnitude improvement over Model-HC*.
Figure 6.6: Convergence of the training (left) and test set (right) likelihoods while optimizing Model-14C parameters with the EM algorithm. Training likelihood characterizes how well the model can learn from observations while testing likelihood characterizes how well the learned model correlates with the generating model. The two cannot be meaningfully compared (as the low training likelihood is mostly the result of a larger number of member sequences) but can be used to compare different models as in Section 6.2.4.1.

Figure 6.7: The likelihood of 113 models whose randomly selected initial parameters were optimized using the EM algorithm.
6.2.4.2 Inference Properties of the Intentional Model

To test the inference algorithms, the defending goalie in the M-ON-N ATTACK scenario is given the probabilistic world model to use as an interpreter of its observations. This allows the goalie to recover likely states of the world, including the hypothesized intentions and joint intentions of the attackers as well as predictions about future states. The two most important characteristics of this inference are its complexity and its accuracy.

The first concern is whether the agent can perform the filtering within the resource constraints of its dynamic task. To test this proposition, a set of observations can be considered concurrently by several different filtering algorithms, and their execution times compared. The different algorithms correspond to the exact inference described in Section 4.5.1 in comparison with several parameterizations of the approximate inference described in Section 4.5.3 (i.e., with different values of \(N_{\psi}\)). The results are presented in Figure 6.8.

![Inference Complexity Graph](image)

Figure 6.8: The time complexity of exact and approximate inference algorithms using the Model-Best model of the 2-ON-0 ATTACK scenario.

Because the filtering computation is performed after each percept is received, it is clear that only the approximate algorithms with less than approximately \(N_{\psi} = 250\) samples have sufficiently quick inference to incorporate into the agent's deliberation process.\(^2\) This conclusion, however, is greatly dependent upon the model parameters. If the model is such that individual or joint intentions terminate frequently, then the inference complexity increases.

\(^2\)Although the agent operates at approximately 10 Hertz, it must act less than 50 milliseconds after sensation in some cases, due to restricted visual perception.
The second concern is the accuracy of the inferred values. Discussing inference accuracy is meaningless without some measure of ground truth and, although there can generally be no ground truth concerning the hypothesized characteristics of the model (for example, the mental attitudes of the observed agents) the model as a whole can be evaluated on two conceptual levels.

1. Evaluation based on the model's prediction of concrete and verifiable notions such as agent actions, elements of the world state, and percepts.

2. Correspondence, particularly with concern to the hypothesized variables, with the designer's intuitive expectations.

The first of the two notions was primarily discussed in Section 6.2. The inference and — more importantly — the prediction of correct actions and intentions is part of the extended, intentional model and will be considered in this section.

For the inferred actions of observed agents, ground truth is obtained by monitoring the SoccerServer program during the scenario to recover a sequence of agent actions, $A_{0:T}^{1:p}$, corresponding to the processed observations. These actions, thereafter considered true, are compared to the predictions provided by the model. For example, a confusion matrix represents the expected probability of correct and incorrect predictions. Specifically, each element of the matrix, $c_{ij}$, represents the probability of predicting action $j$ for some time $\tau > t$, when the agent then performs action $i$ at time $\tau^3$. Two such matrices are shown in Tables 6.4 and 6.5, both for $\tau = t + 1$ (i.e., a one time-step prediction).

Perhaps more important is the inference of variables whose value has longer-term significance, like individual and group intentions. Such inferences are likely to be more useful to an observer's planning process. However, there is no rigorous method of testing these inferences since such variables are hypothesized and are generally hidden even to the experimenter. Nonetheless, the inferences (shown in Figure 6.9) can be analyzed in a hypothetical, qualitative sense. An example sequence was observed by the defending goalie, which made the inferences in Figure 6.9. The important key frames are described in Figure 6.10.

---

3Under the assumption that inferences across time are independent of each other.
Table 6.4: The action confusion matrix produced by Model-HC making inferences from a test set of sequences. For the Move, Shoot, and Dribble actions, the most probable prediction is observed on average; but when an agent performs the Pass action, the observer is expecting it to dribble on average; and when an agent performs the None action, the observer is expecting movement on average. In other words, this model does not capture the rule that induces an agent to pass the ball, as opposed to dribble it.

<table>
<thead>
<tr>
<th>Observed action</th>
<th>Move</th>
<th>Shoot</th>
<th>Pass</th>
<th>Dribble</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>0.4268</td>
<td>0.0062</td>
<td>0.0081</td>
<td>0.2880</td>
<td>0.2709</td>
</tr>
<tr>
<td>Shoot</td>
<td>0.0000</td>
<td>0.6479</td>
<td>0.0374</td>
<td>0.1791</td>
<td>0.1355</td>
</tr>
<tr>
<td>Pass</td>
<td>0.0000</td>
<td>0.2638</td>
<td>0.2253</td>
<td>0.4835</td>
<td>0.0274</td>
</tr>
<tr>
<td>Dribble</td>
<td>0.0000</td>
<td>0.0611</td>
<td>0.0389</td>
<td>0.8920</td>
<td>0.0080</td>
</tr>
<tr>
<td>None</td>
<td>0.5673</td>
<td>0.0009</td>
<td>0.0040</td>
<td>0.0413</td>
<td>0.3865</td>
</tr>
</tbody>
</table>

Table 6.5: The action confusion matrix produced by the Model-Best model making inferences from a test set of sequences, which is similar to that of the Model-HC*. On the whole, the optimized model's are able to predict actions with greater accuracy (as evidenced by the higher likelihood) but more importantly, the correct prediction of individual actions has been greatly improved in all cases to the point that the most probable prediction is always actually effected by the agent under observation. The exception is the None action, which conforms to the statement if it is considered a Move of zero distance.

<table>
<thead>
<tr>
<th>Observed action</th>
<th>Move</th>
<th>Shoot</th>
<th>Pass</th>
<th>Dribble</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>0.6338</td>
<td>0.0002</td>
<td>0.0051</td>
<td>0.1081</td>
<td>0.2528</td>
</tr>
<tr>
<td>Shoot</td>
<td>0.0079</td>
<td>0.7698</td>
<td>0.0815</td>
<td>0.1408</td>
<td>0.0000</td>
</tr>
<tr>
<td>Pass</td>
<td>0.1091</td>
<td>0.0005</td>
<td>0.5832</td>
<td>0.2747</td>
<td>0.0325</td>
</tr>
<tr>
<td>Dribble</td>
<td>0.3286</td>
<td>0.0124</td>
<td>0.0087</td>
<td>0.5840</td>
<td>0.0662</td>
</tr>
<tr>
<td>None</td>
<td>0.5225</td>
<td>0.0007</td>
<td>0.0025</td>
<td>0.0189</td>
<td>0.4554</td>
</tr>
</tbody>
</table>
Figure 6.9: The evolution of joint and individual intentions throughout an example sequence, described in Section 6.2.4.2. For the joint intention (top) the solid line represents \texttt{intend.to.pass} and the dashed line represents \texttt{intend.to.score}; For the individual intention (middle for member 1 and bottom for member 2) the solid line represents \texttt{intend.to.intercept}, the dashed line represents \texttt{intend.to.score}, and the gray, dash-dotted line represents \texttt{intend.to.draw}. 
Initially, the ball is at the centre of the pitch and both attackers are on their half. The goalie is not sure whether their joint intention is `intend_to_pass` or `intend_to_score`, and assumes that attacker 1 will try to score (i.e., has the intention `intend_to_score`) while attacker 2 will try to draw (i.e., has the intention `intend_to_draw`).

However, attacker 2 proceeds toward the ball so the goalie’s beliefs are updated so that attacker 2 is believed to `intend_to_intercept` while attacker 1 is believed to `intend_to_draw`.

After attacker 2 has intercepted the ball and as the attackers approach, the goalie becomes increasingly certain that attacker 1 has the individual intention `intend_to_draw` while attacker 2 has the individual intention `intend_to_score`. However, these inferences do not effect group’s joint intention.

At $t = 71$, attacker 1 begins to behave as if it is attempting to score rather than draw the goalie (based on its positioning), even though attacker 2 still has the ball.

Figure 6.10: Notable moments from a particular 2-ON-0 ATTACK example described in detail.
At $t = 80$, attacker 2 passes the ball to attacker 1. This action initiates significant change in the goalie’s beliefs. Attacker 1 intends to intercept the ball, however the goalie’s belief in `intend_to_score` remains high signalling that attacker 1 will likely attempt to score after intercepting the ball. And after passing the ball, attacker 2 becomes less likely to have the `intend_to_score` intention, and more likely to be attempting to draw the goalie away from attacker 1. With attacker 1 in scoring position, and attacker 2 in drawing position, the goalie infers that their joint intention is more likely `intend_to_pass` than `intend_to_score`, although attacker 1 does go on to shoot the ball.

At $t = 90$, attacker 1 shoots the ball. Although the goalie believes that it is likely that the group is attempting to pass before shooting, it also believes that it it most likely that attacker 1 is intending to score itself.

Figure 6.10: Continued.
Chapter 7

Conclusions

This thesis considers a situated observer in a multiple-agent environment. Although the observer may act and may have other goals, an emphasis was placed on its ability to interpret and understand its surroundings, including the actions of other agents, and construct an appropriate internal representation.

Despite this emphasis, the observer is foremost an agent – an actor – with competitive requirements of rational decisions and reactive behaviour. This thesis introduces an adaptive deliberation scheduling algorithm that dynamically optimizes the balance of reactivity and proactivity. The scheduling algorithm was implemented and applied in a team of agents, situated within the RoboCup Simulation League domain. Theoretical and empirical analysis of the scheduling algorithm has shown that the algorithm generalizes previous approaches such as internal and external synchronization; relaxing their fundamental assumptions with improved, adaptive estimates. Further, it outperforms these approaches in varying conditions, exhibiting the desired robustness to a changing environment. Practically, agents using the new algorithm can choose a degree of reaction while making good decisions, approaching the rational ideal.

Based on this significant success, the adaptive deliberation scheduling algorithm was used as the foundation of a constraint-based agent architecture. The architecture controls perception, modelling, deliberation, means-ends reasoning, and action in an asynchronous and interruptible manner. When the scheduler determines an action should be performed, the architecture always has one available. Also, the more resources that are committed, the more appropriate the resulting action will be. The architecture was used to construct a team of eleven soccer-playing agents within the RSL domain, that competed successfully in the Robot World Cup competition proving that proactive behaviour could be attained without sacrificing responsiveness.

Within the controller processes of perception, modelling, deliberation, and
means-ends reasoning, particular emphasis was placed in perception and modelling. The final significant contribution of this thesis is the development and characterization of a probabilistic model of general multiple-agent behaviours, based on the theory of intentionality. The model was used to understand behaviours in a sub-set of the RSL domain: the 2-ON-0 ATTACK scenario. As a result the thesis develops a novel probabilistic model of the RSL agent's perception and dynamics. When used by the agents to interpret observations through Bayesian filtering, the model is shown to produce more accurate inferences of the world state vector than previous techniques. Experiments also explore the inference of the hypothesized intentionality of observed evidence. The full model's parameters can be learned from observed evidence; and the learned model is shown to provide interesting and intuitive predictions concerning the actions and intentions of the observed agents.

7.1 Avenues of Future Work

The probabilistic models of the RSL agent sensors proved very accurate and sufficient to provide robust behaviours in the RSL domain. However, the inferences are approximate and do not utilize all possible worlds. A promising avenue of future work is to explore other approximations to the complex models that may be more accurate. This may involve a more appropriate sampling method, or parametric approximations of the distributions themselves (e.g., piecewise-linear models).

Further, although the ideas introduced in this thesis were developed in a general context, they were only implemented and tested in the RSL domain (and in some cases a sub-set of that domain). Agents in the RSL domain interact in a variety of ways in support of both individual and coordinated activities, but in general tend to be fairly reactive. An important area of future work is to confirm the findings of this thesis with a larger variety of agents and domains. For example, to see how the adaptive deliberation scheduler performs when the environment's dynamics are less structured, or how the characteristics of the behavioural model change when the processes of intentional deliberation are more (or less) applicable to the domain.

Finally, this thesis has tested the ability of the behavioural model to predict the actions of a group of agents. The underlying assumption is that this prediction will aid deliberation and, ultimately, successful action. However, the link between these inferences and the deliberation process was not rigorously explored. For example, the decision process shown in Figure 7.1 formally represents how these inferred values may impact the commitment to an agent's own intentions. These decision processes can be learned, in conjunction with the model, to also optimize the ac-
tion taken at each time step. It is only with such rigorous interpretation that the intentional model’s usefulness to the observer can be concretely confirmed.

Figure 7.1: A partially observable Markov decision process model of the UBCDynamo agent’s planning and intention management process. The model, not fully detailed for clarity, is based on the intentional representation of Chapter 4. Rectangles represent random variables that the agent can directly effect (decisions) and diamonds represent the value or utility of resulting world states.

Figure 7.1: A partially observable Markov decision process model of the UBCDynamo agent’s planning and intention management process. The model, not fully detailed for clarity, is based on the intentional representation of Chapter 4. Rectangles represent random variables that the agent can directly effect (decisions) and diamonds represent the value or utility of resulting world states.
Bibliography


Equation Derivations

This appendix goes through derivations for some equations used in this thesis, and details exactly how the mathematics used in the thesis relates to equations described in the SoccerServer manual.

Rearrangement of the Equations of Dynamics

In the SoccerServer manual, the equations of dynamics are given in the following form [CFH+02]:

\[ x'' = x + v + a + r \]  \hspace{1cm} (7.1)
\[ v'' = \alpha \cdot (v + a + r) \]  \hspace{1cm} (7.2)

A little mathematical wrangling can produce equations that seem more consistent with traditional physics, yet are mathematically equivalent to the dynamics as presented in the manual. This process, which involves using a new interpretation of velocity, was used to derive the equations of dynamics used in this thesis.

This thesis uses the variables \(\bar{x}, \bar{v},\) and \(\frac{1}{m} \bar{f}\) to denote an object's position, velocity, and applied acceleration respectively. These variables are related to the manual variables in the following way:

\[ \bar{x} = x \]  \hspace{1cm} (7.3)
\[ \bar{v} = \bar{v} + \bar{a} + \bar{r} \]  \hspace{1cm} (7.4)
\[ \frac{1}{m} \bar{f} = \bar{a} + \bar{r} \]  \hspace{1cm} (7.5)

It is rather straightforward to transform the equations of dynamics into this variable space. Starting with Equation 7.1,
\[ \bar{x}' = \bar{x}' \]  
\[ = \bar{x} + \bar{v} + \bar{a} + \bar{r} \]  
\[ = \bar{x} + \bar{v} \]  
(7.6)  
(7.7)  
(7.8)

Next, we determine how the new velocity evolves based on Equation 7.2, using a drag coefficient \( \mu = 1 - a \) instead of the decay parameter \( a \).

\[ \bar{v}' = \bar{v}' + \bar{a}' + \bar{r}' \]  
\[ = \alpha \cdot (\bar{v} + \bar{a} + \bar{r}) + \bar{a} + \bar{r}' \]  
\[ = (1 - \mu) \bar{v} + \frac{1}{m} \tilde{f} \]  
(7.9)  
(7.10)  
(7.11)

It is important to note that the noise term, \( \bar{r} \), has been absorbed into the applied force — random noise applied to the object’s trajectory is viewed as an exogenous force applied to the object by the environment.

**Modelling the RSL Visual Sensor**

The derivation of probability distributions that model the visual sensor are based on the following probabilistic interpretation of the ceiling operator:

\[ \Pr(A = a | [A] = [a]) = \begin{cases} 1 & \text{if } [a] \in [0, 1), \\ 0 & \text{else.} \end{cases} \]  
(7.12)

Given this interpretation, we can replace the ceiling operator with a random variable whose distribution is governed by Equation 7.12. For example, Equation 5.11 follows directly from this procedure. Deriving the distribution of possible ranges is the most complicated and is done in detail below as an example:

\[ ||\tilde{y}|| = \frac{1}{10} \left[ 10 \exp \left\{ q \left[ \frac{1}{q} \ln r - \frac{1}{2} \right] \right\} - \frac{1}{2} \right] \]  
\[ = \exp \left\{ q \left[ \frac{1}{q} \ln r - \frac{1}{2} \right] \right\} + A \]  
\[ = \exp \{ \ln ||\tilde{x}|| + B \} + A \]  
\[ = r \exp B + A \]  
(7.13)  
(7.14)  
(7.15)  
(7.16)

\( A \) and \( B \) represent unique random variables that have no relation to any other parameter discussed in this thesis. Their distribution follows directly from Equation 7.12.
\[ \Pr(A = a \mid \|\vec{g}\|) = \begin{cases} 10 & \text{if } a \in \frac{1}{20}(-1, 1], \\ 0 & \text{else.} \end{cases} \quad (7.17) \]

\[ \Pr(B = b \mid \|\vec{g}\|) = \begin{cases} \frac{1}{q} & \text{if } b \in \frac{q}{2}(-1, 1], \\ 0 & \text{else.} \end{cases} \quad (7.18) \]

Given these two hypothesized random variables (A and B) the distribution for the actual range is determined as follows:

\[
\Pr(R = r \mid Y = \vec{g}) = \int da \int de^b \Pr(e^B = e^b, A = a \| \|\vec{g}\|\|) \Pr(r \mid A = a, e^B = e^b, \|\vec{g}\|) \]

\[ = \int da \int_{e^b}^{e^\frac{q}{2}} de^b q e^{-b} \Pr(A = a \| \|\vec{g}\|\|) \Pr(r \mid A = a, e^B = e^b, \|\vec{g}\|) \]

\[ = \int_{e^{-\frac{q}{2}}}^{e^\frac{q}{2}} da \int_{e^{-\frac{q}{2}}}^{e^\frac{q}{2}} de^b q e^{-b} \Pr(A = \|\vec{g}\| - e^b r \| \|\vec{g}\|\|) \]

\[ = \int \min\left\{ e^\frac{q}{2}, \frac{\|\vec{g}\|}{r} + \frac{1}{20} r \right\} de^b 10 q e^{-b} \]

\[ = 10 q \ln \frac{\max\left\{ e^{-\frac{q}{2}}, \frac{\|\vec{g}\|}{r} - \frac{1}{20} r \right\}}{\min\left\{ e^{\frac{q}{2}}, \frac{\|\vec{g}\|}{r} + \frac{1}{20} r \right\}} \quad (7.23) \]

**Deriving a Stochastic Transition Model for RSL Objects**

This section outlines the procedure of generalizing the equations of dynamics (Equations 2.17 and 2.18) into a probabilistic state transition model. Starting with the equations of dynamics in the form

\[ \bar{x}_{t+1} = \bar{x}_t + \bar{v}_t = \bar{x}_t + (1 - \mu)\bar{v}_{t-1} + \frac{1}{m} f_t \quad (7.24) \]

the state transition model can then be expanded by hypothesizing about the object’s previous velocity and current applied force:

\[
\Pr(X_{t+1} \mid X_t, Y_{0:t}) = \int_{V_{t-1}} \Pr(V_{t-1} \mid X_t, Y_{0:t}) \int_{F_t} \Pr(F_t \mid X_t, Y_{0:t}, V_{t-1}) \times \\
\Pr(X_{t+1} = \bar{x}_{t+1} \mid X_t, Y_{0:t}, V_{t-1}, F_t) \quad (7.25) \]
\[
= \int_{F_t} \Pr(F_t = \tilde{f} | X_t = x_t, Y_{0:t}, V_{t-1} = (1 - \mu)^{-1}(x_{t+1} - x_t - \frac{1}{m} \tilde{f}_t)) \times \\
\Pr(V_{t-1} = (1 - \mu)^{-1}(x_{t+1} - x_t - \frac{1}{m} \tilde{f}_t) | X_t = x_t, Y_{0:t}) \, df_t \quad (7.26)
\]

It is likely that the probability of the object's decayed previous velocity will be known. For example, the agent's own distribution is perceived using Equation 2.20. The applied force probability, however, is more intricate.

### Computing Optimal Trajectories

Computing an optimal trajectory involves consideration of how an object's momentum will evolve into the future. Fortunately, we have accurate equations of dynamics, so this can be done well. For a player, we assume that it will follow the optimal strategy of turning an appropriate amount and then linearly accelerating until it reaches its goal. Accordingly, we have the following predictions:

\[
\begin{align*}
\tilde{x}' &= \tilde{x} + \tilde{v} \\
\tilde{x}'' &= \tilde{x} + \tilde{v} + (1 - \mu)\tilde{v} + \frac{1}{m} \tilde{f} \\
\tilde{x}''' &= \tilde{x} + \tilde{v} \cdot (1 + 1 - \mu + (1 - \mu)^2) + \frac{1}{m} \tilde{f} \\
\vdots &
\end{align*}
\]

Observing the above predictions, a clear pattern emerges, giving us the following prediction equations:

\[
\begin{align*}
\tilde{x}(t + T) &= \tilde{x}(t) + \left( \sum_{i=1}^{T-1} (1 - \mu)^i \right) \tilde{v}(t) + \frac{1}{m} \left( \sum_{j=1}^{T-1} j(1 - \mu)^{T-j-1} \right) \tilde{f} \quad (7.27) \\
\tilde{v}(t + T) &= (1 - \mu)^T \tilde{v}(t) + \frac{1}{m} \left( \sum_{k=0}^{T-1} (1 - \mu)^k \right) \tilde{f} \quad (7.28)
\end{align*}
\]
Glossary of Variables

World state variables for the RSL

The notation for the world state variables follows the following design: $\bar{x}$, $\bar{v}$, $\bar{\theta}$, and $\phi$ represent an object's position, velocity, heading, and head direction respectively. The specific object addressed is indicated as a superscript to that base variable. For example, the RSL world state is composed of the following variables:

- $\bar{x}^a, X^a$: The agent's own position, and a random variable over all possible agent positions.
- $\bar{x}^{pi}, X^{pi}$: The position of player $i$, and a random variable over all possible positions for that player.
- $\bar{x}^b, X^b$: The ball's position, and a random variable over all possible ball positions.
- $\bar{v}^a, V^a$: The agent's own velocity, and a random variable over all possible agent velocities.
- $\bar{v}^{pi}, V^{pi}$: The velocity of player $i$, and a random variable over all possible velocities for that player.
- $\bar{v}^b, V^b$: The ball's velocity, and a random variable over all possible ball velocities.
- $\bar{\theta}^a, \Theta^a$: The agent's own heading, and a random variable over all possible agent headings.
- $\bar{\theta}^{pi}, \Theta^{pi}$: The heading of player $i$, and a random variable over all possible headings for player $i$.
- $\phi^a, \Phi^a$: The agent's own head direction (relative to its body), and a random variable over all possible agent head directions.
- $\phi^{pi}, \Phi^{pi}$: The head direction of player $i$, and a random variable over all possible head directions for player $i$. 
\( \bar{x}, X \) The conglomeration of all of the above variables (and random variables) into the world state vector.

When dynamics need be explicit, a subscript to the above variables denotes the time step that the variable applies to.

Observations of the world state

In general, an observation of a particular variable is denoted using the variable \( y \) with the observed variable placed in superscript. For example,

\[ y^{\theta_a} \] is an observation of the agent's own head orientation, and
\[ y^{v_2} \] is an observation of the player 2's velocity.

Variables modelling agent behaviours

The following variables are used to model the behaviours of other agents (i.e., not the observer) within an environment. The model, described in detail in Chapter 4, characterizes a group of \( p \) agents acting in an environment.

\( A_q^a \) A random variable addressing the action taken by observed agent \( q \). This is a multi-nomial variable encompassing the various types of action the agent may perform (for example, Dash, Turn, and Kick in the RSL). \( |A| \) denotes the number of actions modelled, and the domain is the same for all observed agents.

\( A_i(b) \) The modelled agent's action policy, stochastically providing an agent's action, given its current intention being \( i \) and current beliefs being \( b \). All observed agents have the same policy. See Section 4.4.

\( B^q \) A random variable addressing the beliefs of an observed agent \( q \). The domain of \( B^q \) is the same as the world state, and the same for each observed agent.

\( \bar{\beta} \) A parameter that specifies the dependence of an observed agent's belief, \( B^q \), on the observer's belief. See Equation 4.1.
I^q \quad A random variable addressing the individual intentions of an observed agent \( q \). This is a multi-nomial variable whose domain size, \(|I|\), is the same for each observed agent and denotes the number of distinct propositions the observed agents may intend.

\( I_j(b) \quad \) Represents all agents' individual intention commitment policy. Specifically, it stochastically provides the next individual intention, \( I^q \), given the agent's current belief being \( b \), and the group's joint intention being \( j \). However, a new intention can only be adopted if the previous intention is satisfied or abandoned. See Section 4.2.1.

S^q \quad A random variable addressing the satisfaction or abandonment of agent \( q 's \) current individual intention. If \( S^q = T \), then agent \( q 's \) intention has been satisfied or abandoned. Otherwise, it persists.

\( S_i(b, \tau) \quad \) Represents, indirectly, the sets of propositions that constitute each intention. Specifically, it stochastically determines whether an agent's individual intention is satisfied or abandoned if it has intention \( i \) and beliefs \( b \).

J \quad A random variable addressing the joint intentions of a group of observed agents. This is a multi-nomial variable whose domain size, \(|J|\), denotes the number of distinct propositions the observed group may intend.

\( J(b_1, p) \quad \) Represents the group's joint intention commitment policy. Specifically, it stochastically provides the next joint intention, \( J \), given the group's current joint belief being \( b_1, p \). However, a new joint intention can only be adopted if the previous joint intention is satisfied or abandoned. See Section 4.2.1.

T \quad A random variable addressing the satisfaction or abandonment of the group of agent's current joint intention. If \( T = T \), then the group's joint intention has been satisfied or abandoned. Otherwise, it persists.

\( T_j(b_1, p) \quad \) Represents, indirectly, the sets of propositions that constitute each joint intention. Specifically, it stochastically determines whether the group's joint intention is satisfied or abandoned if it has joint intention \( j \) and joint beliefs \( b_1, p \).

Q \quad In general, this random variable denotes the set of hidden random variables contained within a particular model. For the model of Chapter 4, \( Q \overset{\text{def}}{=} \{J, I^p, A^p, X, B^p, S^p, T\} \)

C, \( \Psi \quad \) A separation of the hidden random variables in the model of Chapter 4 for the purpose of describing properties of an inference algorithm. \( C_t \overset{\text{def}}{=} \{J_t, I^p_t, A^p_t, X_t\} \). \( \Psi_t \overset{\text{def}}{=} \{T_t, S^p_t, B^p_t\} \). See Section 4.5.
A parameter describing the initial probability distribution over the hidden random variables of the model.

**Adaptive resource scheduling**

Most of the variables in this area deal with various time periods related to the internal and external clocks. These details are described mostly in Section 5.1.

\(\Delta, \Delta_{\text{max}}\) A random variable that addresses the amount of delay in a message delivered across the communication channel. The message may be a command from the agent to the environment, or stimuli from the environment to the agent. \(\Delta_{\text{max}}\) is a parameter that specifies the largest possible delay.

\(t^*, \hat{t}^*\) The optimal time to command action within the agent’s cycle with respect to the *external* clock, and the agent’s current estimate of the optimal external time to command action respectively.

\(\tau, \hat{\tau}\) The temporal offset between the internal clock and the external clock, and the agent’s current estimate of the temporal offset between these two clocks respectively.

\(t_{\text{prop}}^{(i)}\) The perception time of the \(i\)th previous proprioception with respect to the internal clock.

\(n_e\) The number of action commands that have been effected. See Section 5.1.3.

\(P, C\) Sets of pending and confirmed action commands, respectively.

**SoccerServer parameters**

See [Nod95, CFH+02] for a more complete listing of SoccerServer parameters.

\(\mu\) The coefficient of friction for a particular object. See Equation 2.18, for example.
$q$ A quantization factor used to corrupt stimuli using Equation 2.21.