Multi-agent actions under uncertainty: situation calculus, discrete time, plans and policies*

David Poole
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
poole@cs.ubc.ca
http://www.cs.ubc.ca/spider/poole

April 23, 1997

Abstract

We are working on a logic to combine the advantages of first-order logic, but using Bayesian decision theory (or more generally game theory) as a basis for handing uncertainty. This forms a logic for multiple agents under uncertainty. These agents act asynchronously, can have their own goals, have noisy sensors, and imperfect effectors. Recently we have developed the independent choice logic that incorporates all of these features. In this paper we discuss two different representations of time within this framework: the situation calculus and what is essentially the event calculus. We show how they both can be used, and compare the different ontological commitments made by each. Uncertainty is handled in terms of a logic which allows for independent choices and a logic program that gives the consequences of the choices. There are probabilities over the choices by nature. As part of the

*This work was supported by Institute for Robotics and Intelligent Systems, Project IC-7 and Natural Sciences and Engineering Research Council of Canada Operating Grant OGPOO44121.

consequences are a specification of the utility of (final) states. In the situation calculus, agents adopt programs and programs lead to situations in possible worlds (which correspond to the possible outcomes of complete histories); given a probability distribution over possible worlds, we can get the expected utility of a program. In the event calculus view, actions are propositions and agents adopt policies which are logic programs to imply what the agent will do based on what it observes. Again the expected value of a policy can be computed. The aim is to choose the plan or policy that maximizes the expected utility. This paper overviews both approaches, and explains why I think the event calculus is the most promising approach.

1 Introduction

1.1 Logic and Uncertainty

There are many normative arguments for the use of logic in AI (see e.g., Nilsson 1991, Poole, Mackworth & Goebel 1997). These arguments are usually based on reasoning with symbols with an explicit denotation, allowing relations amongst individuals, and quantification over individuals. This is often translated as needing (at least) the first-order predicate calculus. However, the first-order predicate calculus has very primitive mechanisms for handling uncertainty.

There are also normative arguments for Bayesian decision theory and game theory (see e.g., Von Neumann & Morgenstern 1953, Savage 1972) for handling uncertainty. These are based on theorems that show that, under certain reasonable assumptions, rational decision makers will act as though they are using probabilities and utilities and maximizing their expected utilities. Game theory (Von Neumann & Morgenstern 1953, Myerson 1991, Fudenberg & Tirole 1992) is the extension of Bayesian decision theory to include multiple intelligent agents.

We would like to combine the advantages of logic with those of Bayesian decision/game theory. The independent choice logic (ICL) (Poole 1997) is designed with the goal of including the advantages of logic, but handling *all* uncertainty using Bayesian decision or game theory.

The idea is to not use disjunction, but rather to allow agents, including nature, to make choices from a choice space, and use a restricted underlying logic to specify the consequences of the choices. To start off without disjunction, we can adopt acyclic logic programs (Apt & Bezem 1991) as the underlying logical formalism. What is interesting is that simple logic programming solutions to the frame problem (see e.g., Shanahan 1997, Chapter 12) seem to be directly transferrable to the

ICL which has more sophisticated mechanisms for handling uncertainty than the predicate calculus. I would even dare to venture that the main problems with formalizing action within the predicate calculus arise because of the inadequacies of disjunction to represent the sort of uncertainty we need.

When mixing logic and probability, one can extend a rich logic with probability, and have two sorts of uncertainty: that uncertainty from the probabilities and that from disjunction in the logic (Bacchus 1990, Halpern & Tuttle 1993). An alternative that is pursued in the independent choice logic is to have all of the uncertainty in terms of probabilities. The underlying logic is restricted so that there is no uncertainty in the logic[1]; every set of sentences has a unique model. In particular we choose the logic of acyclic logic programs under the stable model semantics; this is a practical language with the unique model property. All uncertainty is handled by what can be seen as independent stochastic mechanisms. A deterministic logic program that gives the consequences of the agent's choices and the random outcomes. In this manner we can get a simple mix of logic and Bayesian decision theory (Poole 1997).

1.2 Actions and Uncertainty

The combination of decision or game theory and planning (Feldman & Sproull 1975) is very appealing. The general idea of planning is to construct a sequence of steps, perhaps conditional on observations that solves a goal. In decision-theoretic planning, this is generalised to the case where there is uncertainty about the environment and we are concerned, not only with solving a goal, but what happens under any of the contingencies. Goal solving is extended to the problem of maximizing the agent's expected utility, where the utility is an arbitrary function of the final state (or the accumulation of rewards received earlier). Moreover in the multi-agent case, each agent may have a different utility, they carry out actions asynchronously, and the actions of all of the agents affect the utility for each agent. Moreover there is uncertainty about the environment, and it is often optimal for an agent to act stochastically. It is our goal to write a logic that allows for practical reasoning for domains that include multiple agents, utilities, uncertainty and stochastic actions.

Within the independent choice logic, we have considered both the situation cal-

[1] We cannot conclude $a \vee b$ without being able to conclude a or b. We also want to be able to conclude either a or $\neg a$ for every closed formula a. Note that this is a property of the underlying logic, not a property of the ICL.

culus (Poole 1996), discrete time (Poole 1997), and continuous time (Poole 1995). In this paper we compare the situation calculus (McCarthy & Hayes 1969) and the discrete time frameworks in this framework.

Recently there have been claims made that Markov decision processes (MDPs) (Puterman 1990) are the appropriate framework for developing decision theoretic planners (e.g., Boutilier, Dearden & Goldszmidt 1995). MDPs, and dynamical system in general (Luenberger 1979) are based on the notion of a **state**: what is true at a time such that the past at that time can only affect the future from that time by affecting the state. In terms of probability: the future is independent of the past given the state. This is called the Markov property. In the most general framework the agents individually have states (these are called belief states) and idea is to construct a state transition function that specify how the agent's belief state is updated from its previous belief state and its observations, and a command function (policy) that specifies what the agent should do based on its observations and belief state (Poole et al. 1997, Chapter 12). In fully observable MDPs, the agent can observe the actual state and so doesn't need belief states. In partially observable MDPs, the belief state is a probability distribution over the actual states of the system, and the state transition function is given by Bayes' rule. In between these are agents that have limited memory or limited reasoning capabilities.

For a representation for time, actions and states we can adopt a number of different representations. One is to consider robot programs as policies (Poole 1995) where actions are propositions, and agents adopt logic programs that specify what they will do based on their observations and belief state. An alternative is to represent actions in the situation calculus, where the agents have conditional plans (Poole 1996). Both are discussed here.

All of the uncertainty in our rules is relegated to independent choices as in the independent choice logic (Poole 1997) (an extension of probabilistic Horn abduction (Poole 1993) to include multiple agents and negation as failure).

2 The Independent Choice Logic

An independent choice space theory is make of two principal components: a choice space that specifies what choice can be made, and a set of facts that specifies what follows from the choices.

Definition 2.1 A **choice space** is a set of sets of ground atomic formulae, such that if χ_1, and χ_2 are in the choice space, and $\chi_1 \neq \chi_2$ then $\chi_1 \cap \chi_2 = \{\}$. An element

of a choice space is called a **choice alternative** (or sometimes just an alternative). An element of a choice alternative is called an **atomic choice**.

Definition 2.2 Given choice space $\mathcal{C}$, a **selector function** is a mapping $\tau : \mathcal{C} \rightarrow \cup\mathcal{C}$ such that $\tau(\chi) \in \chi$ for all $\chi \in \mathcal{C}$. The **range** of selector function τ, written $\mathcal{R}(\tau)$ is the set $\{\tau(\chi) : \chi \in \mathcal{C}\}$. The range of a selector function is called a **total choice**. In other words, a total choice is a selection of one member from each element of $\mathcal{C}$.

Definition 2.3 The **Facts**, $\mathcal{F}$, are sentences in some base logic with the following two properties:

- For every selector function τ, the set of sentences $\mathcal{F} \cup \mathcal{R}(\tau)$ is definitive on every proposition; that is, for every proposition a either $\mathcal{F} \cup \mathcal{R}(\tau) \mathrel{|\!\sim} a$ or $\mathcal{F} \cup \mathcal{R}(\tau) \mathrel{|\!\sim} \neg a$, where $\mathrel{|\!\sim}$ is logical consequence in the underlying logic.
- For all atomic choices $a \in \cup\cup\mathcal{C}$, $\mathcal{F} \cup \mathcal{R}(\tau) \mathrel{|\!\sim} a$ iff $a \in \mathcal{R}(\tau)$.

The restriction really means that the base logic contains no uncertainty. All uncertainty is handled in the choice space.

The semantics of an ICL is defined in terms of possible worlds. There is a possible world for each selection of one element from each alternative. The atoms which follow using the consequence relation from these atoms together with $\mathcal{F}$ are true in this possible world.

Definition 2.4 Suppose we are given a base logic and ICL theory $\langle\mathcal{C}, \mathcal{F}\rangle$. For each selector function τ there is a **possible world** w_τ. We write $w_\tau \models_{\langle\mathcal{C},\mathcal{F}\rangle} f$, read "$f$ is true in world w_τ based on $\langle\mathcal{C}, \mathcal{F}\rangle$", iff $\mathcal{F} \cup \mathcal{R}(\tau) \mathrel{|\!\sim} f$. When understood from context, the $\langle\mathcal{C}, \mathcal{F}\rangle$ is omitted as a subscript of $\models$.

The fact that every proposition is either true or false in a possible world follows from the definitiveness of the base logic.

Note that, for each alternative $\chi \in \mathcal{C}$ and for each world w_τ, there is exactly one element of χ that's true in w_τ. In particular, $w_\tau \models \tau(\chi)$, and $w_\tau \not\models \alpha$ for all $\alpha \in \chi - \{\tau(\chi)\}$.

The base logic we use is that of acyclic logic programs (Apt & Bezem 1991), such that no atomic choice is in the head of any rule. $A \mathrel{|\!\sim} f$ means f is true in the (unique) stable model of A. This logic is definitive in the above sense and is rich enough to axiomatise many of the domains we are interested in. Note that acyclic logic programs allow recursion, but all recursion must be well founded.

This semantic construction is the core of the ICL. Other components we require for different applications include:

$\mathcal{A}$ is a finite set of agents. There is a distinguished agent 0 called "nature".

$controller$ is a function from $\mathcal{C} \rightarrow \mathcal{A}$. If $controller(\chi) = a$ then agent a is said to control alternative χ. If $a \in \mathcal{A}$ is an agent, the set of alternatives controlled by a is $\mathcal{C}_a = \{\chi \in \mathcal{C} : controller(\chi) = a\}$. Note that $\mathcal{C} = \bigcup_{a \in \mathcal{A}} \mathcal{C}_a$.

P_0 is a function $\cup\mathcal{C}_0 \rightarrow [0,1]$ such that $\forall \chi \in \mathcal{C}_0, \sum_{\alpha \in \chi} P_0(\alpha) = 1$. That is, for each alternative controlled by nature, P_0 is a probability measure over the atomic choices in the alternative.

When each agent (other than nature) makes a choice (possibly stochastic) from each alternative it controls, we can determine the probability of any proposition. The **probability** of a proposition is defined in the standard way. For a finite choice space[2], the probability of any proposition is the sum of the probabilities of the worlds in which it is true. The probability of a possible world is the product of the probabilities of the atomic choices that are true in the world. That is, the atomic choices are (unconditionally) probabilistically independent. Poole (1993) proves that such independent choices together with an acyclic logic program can represent any finite probability distribution. Moreover the structure of the rule-base mirrors the structure of Bayesian networks (Pearl 1988)[3]. Similarly we can define the **expectation** of a function that has a value in each world, as the value averaged over all possible worlds, weighted by their probability.

See Poole (1997) for more details on the ICL.

3 The Situation Calculus and the ICL

In this section we sketch how the situation calculus can be embedded in the ICL. What we must remember is that we only need to axiomatise the deterministic aspects in the logic programs; the uncertainty is handled separately. What gives us confidence that we can use simple solutions to the frame problem, for example, is that every statement that is a consequence of the facts that doesn't depend on the atomic choices is true in every possible world. Thus, if we have a property that

[2]Poole (1993) shows how to model the case where the choice space isn't finite. Essentially we need measurable sets of worlds.

[3]This mapping also lets us see the relationship between the causation that is inherent in Bayesian networks (Pearl 1995) and that of the logical formalisms. See Poole (1993) for a discussion on the relationship, including the Bayesian network solution to the Yale shooting problem and stochastic variants.

depends only on the facts and is robust to the addition of atomic choices, then it will follow in the ICL; we would hope than any logic programming solution to the frame problem would have this property.

Before we show how to add the situation calculus to the ICL, there are some design choices that need to be made even to consider just single agents.

- In the deterministic case, the trajectory of actions by the (single) agent up to some time point determines what is true at that point. Thus, the trajectory of actions, as encapsulated by the 'situation' term of the situation calculus (McCarthy & Hayes 1969, Reiter 1991) can be used to denote the state, as is done in the traditional situation calculus. However, when dealing with uncertainty, the trajectory of an agent's actions up to a point, does not uniquely determine what is true at that point. What random occurrences or exogenous events occurred also determines what is true. We have a choice: we can keep the semantic conception of a situation (as a state) and make the syntactic characterization more complicated by perhaps interleaving exogenous actions, or we can keep the simple syntactic form of the situation calculus, and use a different notion that prescribes truth values. We have chosen the latter, and distinguish the 'situation' denoted by the trajectory of actions, from the 'state' that specifies what is true in the situation. In general there will be a probability distribution over states resulting from a set of actions by the agent. It is this distribution over states, and their corresponding utility, that we seek to model.

  This division means that agent's actions are treated very differently from exogenous actions that can also change what is true. The situation terms define only the agent's actions in reaching that point in time. The situation calculus terms indicate only the trajectory, in terms of steps, of the agent and essentially just serve to delimit time points at which we want to be able to say what holds. This is discussed further in Section 5.

- None of our representations assume that actions have preconditions; all actions can be attempted at any time. The effect of the actions can depend on what else is true in the world. This is important because the agent may not know whether the preconditions of an action hold, but, for example, may be sure enough to want to try the action.

- When building conditional plans, we have to consider what we can condition these plans on. We assume that the agent has passive sensors, and that

it can condition its actions on the output of these sensors. We only have one sort of action, and these actions only affect 'the world' (which includes both the robot and the environment). All we need to do is to specify how the agent's sensors depend on the world. This does not mean that we cannot model information-producing actions (e.g., looking in a particular place) — these information producing actions produce effects that make the sensor values correlate with what is true in the world. The sensors can be noisy; the value they return does not necessarily correspond with what is true in the world (of course if there was no correlation with what is true in the world, they would not be very useful sensors).

Before we introduce the probabilistic framework we present the situation calculus (McCarthy & Hayes 1969). The general idea is that robot actions take the world from one 'situation' to another situation. We assume there is a situation s_0 that is the initial situation, and a function $do(A, S)$[4] that given action A and a situation S returns the resulting situation. An agent that knows what it has done, knows what situation it is in. It however does not necessarily know what is true in that situation. The robot may be uncertain about what is true in the initial situation, what the effects of its actions are and what exogenous events occurred.

3.1 The ICL_{SC}

Within the ICL we can use the situation calculus as a representation for change. Within the logic, there is only one agent, nature, who controls all of the alternatives. These alternatives thus have probability distributions over them. The probabilities are used to represent our ignorance of the initial state and the outcomes of actions. We can then use the situations to reflect the "time" at which some fluents are true or not.

We model all randomness as independent stochastic mechanisms, such that an external viewer that knew the initial state (i.e., what is true in the situation s_0), and knew how the stochastic mechanisms resolved themselves would be able to predict what was true in any situation. Given a probability distribution over the initial state and the stochastic mechanisms, we have a probability distribution over the effects of actions. This is modelled by having the mechanisms as atomic choices controlled by nature (and so with a probability distribution).

[4]We assume the Prolog convention that variables are in upper case and constants are in lower case. Free variables in formulae are considered to be universally quantified with the widest scope.

We use logic to specify the transitions specified by actions and thus what is true in a situation. What is true in a situation depends on the action attempted, what was true before and what stochastic mechanism occurred. A fluent is a predicate (or function) whose value in a world depends on the situation; we use the situation as the last argument to the predicate (function). We assume that for each fluent we can axiomatise in what situations it is true based on the action that was performed, what was true in the previous state and the outcome of the stochastic mechanism.

Note that a possible world in this framework corresponds to a complete history. A possible world specifies what is true in each situation. In other words, given a possible world and a situation, we can determine what is true in that situation.

Example 3.1 We can write rules such as, the robot is carrying the key after it has (successfully) picked it up:

$$\begin{aligned} &carrying(key, do(pickup(key), S)) \leftarrow \\ &\quad at(robot, Pos, S) \wedge \\ &\quad at(key, Pos, S) \wedge \\ &\quad pickup\_succeeds(S). \end{aligned}$$

Here $pickup\_succeeds(S)$ is true if the agent would succeed if it picks up the key and is false if the agent would fail to pick up the key. The agent typically does not know the value of $pickup\_succeeds(S)$ in situation S, or even the position of the key. We would expect that each ground instance of $pickup\_succeeds(S)$ would be an atomic choice. That is

$$\forall S \; \{pickup\_succeeds(S), pickup\_fails(S)\} \in \mathcal{C}_0$$

$P_0(pickup\_succeeds(S))$ reflects how likely it is that the agent succeeds in carrying the key given that it was at the same position as the key and attempted to pick it up.

The general form of a frame axiom specifies that a fluent is true after a situation if it were true before, and the action were not one that undid the fluent, and there was no mechanism that undid the fluent.[5]

[5]This is now a reasonably standard logic programming solution to the frame problem (Shanahan 1997, Chapter 12), (Apt & Bezem 1991). It is essentially the same as Reiter's (1991) solution to the frame problem. It is closely related to Kowalski's (1979) axiomatization of action, but for each proposition, we specify which actions are exceptional, whereas Kowalski specifies for every every action which propositions are exceptional. Kowalski's representation could also be used here.

Example 3.2 For example, an agent is carrying the key as long as the action was not to put down the key or pick up the key, and the agent did not accidentally drop the key while carrying out another action:

$$\begin{aligned} &carrying(key, do(A, S)) \leftarrow \\ &\quad carrying(key, S) \wedge \\ &\quad A \neq putdown(key) \wedge \\ &\quad A \neq pickup(key) \wedge \\ &\quad keeps\_carrying(key, S). \end{aligned}$$

$keeps\_carrying(key, S)$ may be something that the agent does not know whether it is true — there may be a probability that the agent will drop the key. If dropping the key is independent at each situation, we can model this as:

$$\forall S \, \{keeps\_carrying(key, S), drops(key, S)\} \in \mathcal{C}_0$$

This thus forms a stochastic frame axiom.

3.2 Axiomatising utility

Given the notion of an ICL_{SC} theory, we can write rules for utility. Assume the utility depends on the situation that the robot ends up in and the possible world. In particular we allow for rules that imply $utility(U, S)$, which is true in a possible world if the utility is U for situation S in that world. The utility depends on what is true in the state defined by the situation and the world — thus we write rules that imply $utility$. In order to make sure that we can interpret these rules as utilities we need to have utility being functional: for each situation S, and for each possible world w_τ, there exists a unique U such that $utility(U, S)$ true in w_τ. If this is the case we say the theory is **utility complete**. Ensuring utility completeness can be done locally; we have to make sure that the rules for utility cover all of the cases and there are not two rules that imply different utilities whose bodies are compatible.

Example 3.3 Suppose the utility is the sum of the 'prize' plus the remaining resources:

$$\begin{aligned} &utility(R + P, S) \leftarrow \\ &\quad prize(P, S) \wedge \\ &\quad resources(R, S). \end{aligned}$$

The prize depends on whether the robot reached its destination or it crashed. No matter what the definition of any other predicates is, the following definition of $prize$ will ensure there is a unique prize for each world and situation:

$$prize(-1000, S) \leftarrow crashed(S).$$
$$prize(1000, S) \leftarrow in\_lab(S) \wedge \neg crashed(S).$$
$$prize(0, S) \leftarrow \neg in\_lab(S) \wedge \neg crashed(S).$$

The resources used depends not only on the final state but on the route taken. To model this we make $resources$ a fluent, and like any other fluent we axiomatise it:

$$resources(200, s_0).$$
$$resources(R - Cost, do(goto(To, Route), S)) \leftarrow$$
$$\quad at(robot, From, S) \wedge$$
$$\quad path(From, To, Route, Cost) \wedge$$
$$\quad resources(R, S).$$
$$resources(R - 10, do(A, S)) \leftarrow$$
$$\quad \neg gotoaction(A) \wedge$$
$$\quad resources(R, S).$$
$$gotoaction(goto(A, S)).$$

Here we have assumed that non-goto actions cost 10, and that paths have costs. Paths and their costs can be axiomatised using $path(From, To, Route, Risky, Cost)$ that is true if the path from $From$ to To via $Route$ has cost $Cost$.

3.3 Axiomatising Sensors

We also need to axiomatise how sensors work. We assume that sensors are passive; this means that they receive information from the environment, rather than "doing" anything; there are no sensing actions. This seems to be a better model of actual sensors, such as eyes or ears, and makes modelling simpler than when sensing is an action. So called "information producing actions" (such as opening the eyes, or performing a biopsy on a patient, or exploding a parcel to see if it is (was) a bomb) are normal actions that are designed to change the world so that the sensors correlate with the value of interest. Note that under this view, there are no information producing actions, or even informational effects of actions; rather various conditions in the world, some of which are under the robot's control and some of which are not, work together to give varying values for the output of sensors.

A robot cannot condition its action on what is true in the world; it can only condition its actions on what it senses and what it remembers. The only use for sensors is that the output of a sensor depends, perhaps stochastically, on what is true in the world, and thus can be used as evidence for what is true in the world.

Within our situation calculus framework, can write axioms to specify how sensed values depend on what is true in the world. What is sensed depends on the situation and the possible world. We assume that there is a predicate $sense(C, S)$ that is true if C is sensed in situation S. Here C is a term in our language, that represents one value for the output of a sensor. C is said to be **observable**.

Example 3.4 A sensor may be to be able to detect whether the robot is at the same position as the key. It is not reliable; sometimes it says the robot is at the same position as the key when it is not (a false positive), and sometimes it says that the robot is not at the same position when it is (a false negative). Suppose that noisy sensor at_key detects whether the agent is at the same position as the key. Fluent $sense(at\_key, s)$ is true (in a world) if the robot senses that it is at the key in situation s. It can be axiomatised as:

$$
\begin{aligned}
&sense(at\_key, S) \leftarrow \\
&\quad at(robot, P, S) \wedge \\
&\quad at(key, P, S) \wedge \\
&\quad sensor\_true\_pos(S). \\
&sense(at\_key, S) \leftarrow \\
&\quad at(robot, P_1, S) \wedge \\
&\quad at(key, P_2, S) \wedge \\
&\quad P_1 \neq P_2 \wedge \\
&\quad sensor\_false\_pos(S).
\end{aligned}
$$

The fluent $sensor\_false\_pos(S)$ is true if the sensor is giving a false-positive value in situation S, and $sensor\_true\_pos(S)$ is true if the sensor is not giving a false negative in situation S. Each of these could be part of an atomic choice, which would let us model sensors whose errors at different times are independent.

$$
\begin{aligned}
&\forall S \; \{sensor\_true\_pos(S), sensor\_false\_neg(S)\} \in \mathcal{C}_0 \\
&\forall S \; \{sensor\_false\_pos(S), sensor\_true\_neg(S)\} \in \mathcal{C}_0
\end{aligned}
$$

If we had a theory about how sensors break, we could write rules that imply these fluents.

3.4 Conditional Plans

The idea behind the ICL$_{SC}$ is that agents get to choose situations (they get to choose what they do, and when they stop), and 'nature' gets to choose worlds (there is a probability distribution over the worlds that specifies the distribution of effects of the actions).

Agents do not directly adopt situations, they adopt 'plans' or 'programs'. In general these programs can involve atomic actions, conditioning on observations, loops, nondeterministic choice and procedural abstraction (Levesque, Reiter, Lespérance, Lin & Scherl 1996). In this paper we only consider simple conditional plans which are programs consisting only of sequential composition and conditioning on observations (Levesque 1996, Poole 1996)).

An example conditional plan is:

a; if c then b else d; e endIf; g

An agent executing this plan will start in situation s_0, then do action a, then it will sense whether c is true in the resulting situation. If c is true, it will do b then g, and if c is false it will do d then e then g. Thus this plan either selects the situation $do(g, do(b, do(a, s_0)))$ or the situation $do(g, do(e, do(d, do(a, s_0))))$. It selects the former in all worlds where $sense(c, do(a, s_0))$ is true, and selects the latter in all worlds where $sense(c, do(a, s_0))$ is false. Note that each world is definitive on each fluent for each situation. The expected utility of this plan is the weighted average of the utility for each of the worlds and the situation chosen for that world. The only property we need of c is that its value in situation $do(a, s_0)$ will be able to be observed. The agent does not need to be able to determine its value beforehand.

Definition 3.5 A **conditional plan**, or just a **plan**, is of the form

$skip$
A where A is a primitive action
$P; Q$ where P and Q are plans
if C then P else Q endIf
where C is observable; P and Q are plans

Note that "$skip$" is not an action; the $skip$ plan means that the agent does not do anything — time does not pass. This is introduced so that the agent can stop without doing anything (this may be a reasonable plan), and so we do not need an "if C then P endIf" form as well; this would be an abbreviation for "if C then P else $skip$ endIf".

Plans select situations in worlds. We can define a relation:

$trans(P, W, S_1, S2)$

that is true if doing plan P in world W from situation S_1 results in situation S_2. This is similar to the DO macro of Levesque et al. (1996) and the Rdo of Levesque (1996), but here what the agent does depends on what it observes, and what the agent observes depends on which world it happens to be in.

We can define the $trans$ relation in pseudo Prolog as:

$trans(skip, W, S, S).$
$trans(A, W, S, do(A, S)) \leftarrow$
    $primitive(A).$
$trans((P; Q), W, S_1, S_3) \leftarrow$
    $trans(P, W, S_1, S_2) \wedge$
    $trans(Q, W, S_2, S_3).$
$trans((\text{if } C \text{ then } P \text{ else } Q \text{ endIf}), W, S_1, S_2) \leftarrow$
    $W \models sense(C, S_1) \wedge$
    $trans(P, W, S_1, S_2).$
$trans((\text{if } C \text{ then } P \text{ else } Q \text{ endIf}), W, S_1, S_2) \leftarrow$
    $W \not\models sense(C, S_1) \wedge$
    $trans(Q, W, S_1, S_2).$

Now we are at the stage where we can define the expected utility of a plan. The expected utility of a plan is the weighted average, over the set of possible worlds, of the utility the agent receives in the situation it ends up in for that possible world:

Definition 3.6 If our theory is utility complete, the **expected utility** of plan P is[6]:

$$\varepsilon(P) = \sum_{\tau} p(w_\tau) \times u(w_\tau, P)$$

(summing over all selector functions τ on $\mathcal{C}_0$) where

$u(W, P) = U$ if $W \models utility(U, S)$
    where $trans(P, W, s_0, S)$

[6]We need a slightly more complicated construction when we have infinitely many worlds. We need to define probability over measurable subsets of the worlds (Poole 1993), but that would only complicate this presentation.

(this is well defined as the theory is utility complete), and

$$p(w_\tau) = \prod_{\chi_0 \in \mathcal{R}(\tau)} P_0(\chi_0)$$

$u(W, P)$ is the utility of plan P in world W. $p(w_\tau)$ is the probability of world w_τ. The probability is the product of the independent choices of nature. It is easy to show that this induces a probability measure (the sum of the probabilities of the worlds is 1).

4 Independent Choice Logic and Reactive Policies

There is a completely different way to use the ICL to model time and action. Here we can only sketch the idea; see Poole (1997) for details. We only consider discrete time here.

The idea is to made agents and nature in the same way. For the situation calculus axiomatization above, the single agent was treated in a completely different way to nature. Symmetry is important when we consider multiple agents.

We represent time in terms of the integers. The fact that the agent attempted an an action is represented a propositions indexed by time. We can use a predicate $do(A, T)$ that is true if the agent attempted action A at time T. What is true at a time depends on what was true at the previous times and what actions have occurred, and the outcome of stochastic mechanisms. This places actions by the agent at the same level as actions by nature.

There are two parts to axiomatise. The first is to axiomatise the effect of actions, and the second is to specify what an agent will do based on what it observes (i.e., its policy).

To axiomatise the effect of actions, for the discrete time case we can simply write how what is true at one time depends on what was true at the previous time (including what actions occurred). We would write similar axioms to the situation calculus, but indexed by time, and using do as a predicate.

Example 4.1 The axiom for carrying of Example 3.1 can be stated as:

$$\begin{aligned} &carrying(key, T+1) \leftarrow \\ &\quad do(pickup(key), T) \wedge \\ &\quad at(robot, Pos, T) \wedge \\ &\quad at(key, Pos, T) \wedge \\ &\quad pickup\_succeeds(T). \end{aligned}$$

The frame axiom for $carrying$ in Example 3.2 would look like:

$$\begin{aligned}
&carrying(key, T+1) \leftarrow \\
&\quad carrying(key, T) \wedge \\
&\quad \neg do(putdown(key), T) \wedge \\
&\quad \neg do(pickup(key), T) \wedge \\
&\quad keeps_carrying(key, S).
\end{aligned}$$

These don't look very different to the situation calculus axioms!

Similarly axioms for sensing that only refer to a single situation/state, such as those of Example 3.4 would remain the same, but the variables are quantified over times, not situations.

This slight change to the representation of the facts has profound effects on the plans. There are no situations. What an agent does is a set of propositions for different times. Within this framework, it is natural to think in terms of agents adopting policies.

What an agent does depends on what it observes and what it remembers. A **policies** is a logic program that specify what an will do based on what it senses and what it has remembered. For example,

$$\begin{aligned}
&do(pickup(key), T) \leftarrow \\
&\quad sense(at_key, T) \wedge \\
&\quad recall(want_key, T).
\end{aligned}$$

Here $recall$ could be a predicate that represents the internal state of the agent. It can be axiomatised like any other relation.

This rule is a very different to a situation calculus program, because it says that whenever the robot senses it is at a key, and wants it, it should pick it up (as well as doing any other actions that are implied by other rules). In order to implement a situation-calculus type plan using such rules, the robot needs to maintain something like a program counter or continuations. In order for a situation calculus program to implement such rules, it has to loop over a conditional statement that checks the conditions of the rules, and does the appropriate concurrent actions.

With axioms about utility, a policy has a utility in a possible world, and so, by averaging over possible worlds it has an expected utility. The goal is to choose the policy with the highest expected utility.

Within this framework, concurrent actions and multiple agents are easy to represent. The proposed framework here is, like the event calculus, narrative-based

Shanahan (1997) in that it is reasoning about a particular course of events. This is true for each possible world, but we can have a probability distribution over possible worlds. We have a mechanism for allowing multiple agents to choose what events that they can control occur, and to allow a probability distribution over events that nature controls. For each world, we only need to worry about what events are true in that world.

5 Discussion

We have given a (too) brief sketch of two different representations of change for a single agent under uncertainty in the ICL. See Poole (1997) and Poole (1996) for more details.

In some sense the axioms look similar; there is not really much difference between the situation calculus and event calculus axiomatization given here. The major difference is that the robot programs that are natural for the situation calculus are very different from the reactive policies that are natural for the event calculus.

When we extend this to multiple agents, and stochastic actions (as is often needed for agents with limited sensing and communication), the event calculus framework can easily be adapted. Nothing needs to be changed to allow for concurrent actions (the actions by each agent). Each agent adopts its own (private) policy based on what it can sense and what it can do.

Extending the situation calculus version to multiple agents isn't so straightforward. The way we have treated the situation calculus (and we have tried hard to keep it as close to the original as possible) really gives an agent-oriented view of time — the 'situations' in some sense mark particular time points that correspond to the agent completing its actions. Everything else (e.g., actions by nature or other agents) then has to meld in with this division of time. This is even trickier when we realize that when agents have sloppy actuators and noisy sensors, the actions defining the situations correspond to action attempts; the agent doesn't really know what it did it only knows what it attempted and what its sensors now tell it. When there are multiple agents, either there has to be a common clock, we need some master agent which the other agents define their state transition, or complex actions (Reiter 1996, Lin & Shoham 1995). These all mean that the actions need to be carried out lock-step, removing the intuitive appeal of the situation calculus, and making it much closer to the event calculus. The work of Reiter (1996) and Lin & Shoham (1995) assumes a very deterministic world. Not only must the world unfold deterministically, but you must know how it unfolds. This is very different

to the assumptions that hold here, where an agent doesn't even know what it has done, only what it has attempted. The work here may show how to reconcile such "God's view" with stochastic actions and limited sensing.

References

Apt, K. R. & Bezem, M. (1991). Acyclic programs, *New Generation Computing* **9**(3-4): 335–363.

Bacchus, F. (1990). *Representing and Reasoning with Uncertain Knowledge*, MIT Press, Cambridge, Massachusetts.

Boutilier, C., Dearden, R. & Goldszmidt, M. (1995). Exploiting structure in policy construction, *Proc. 14th International Joint Conf. on Artificial Intelligence (IJCAI-95)*, Montreal, Quebec, pp. 1104–1111.

Feldman, J. R. & Sproull, R. F. (1975). Decision theory and artificial intelligence II: The hungry monkey, *Cognitive Science* **1**: 158–192.

Fudenberg, D. & Tirole, J. (1992). *Game Theory*, MIT Press, Cambridge Massachusetts.

Halpern, J. & Tuttle, M. (1993). Knowledge, probability, and adversaries, *Journal of the ACM* **40**(4): 917–962.

Kowalski, R. (1979). *Logic for Problem Solving*, Artificial Intelligence Series, North Holland, New York.

Levesque, H. J. (1996). What is planning in the presence of sensing?, *Proc. 13th National Conference on Artificial Intelligence*, Portland, Oregon, pp. 1139–1146.

Levesque, H. J., Reiter, R., Lespérance, Y., Lin, F. & Scherl, R. B. (1996). GOLOG: A logic programming language for dynamic domains, *Journal of Logic Programming, Special issue on Reasoning about Action and Change* **to appear**.

Lin, F. & Shoham, Y. (1995). Provably correct theories of action, *Journal of ACM* **42**(2): 283–320.

Luenberger, D. G. (1979). *Introduction to Dynamic Systems: Theory, Models and Applications*, Wiley, New York.

McCarthy, J. & Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence, *in* M. Meltzer & D. Michie (eds), *Machine Intelligence 4*, Edinburgh University Press, pp. 463–502.

Myerson, R. B. (1991). *Game Theory: Analysis of Conflict*, Harvard University Press, Cambridge, MA.

Nilsson, N. J. (1991). Logic and artificial intelligence, *Artificial Intelligence* **47**: 31–56.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA.

Pearl, J. (1995). Causal diagrams for empirical research, *Biometrika* **82**(4): 669–710.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks, *Artificial Intelligence* **64**(1): 81–129.

Poole, D. (1995). Logic programming for robot control, *Proc. 14th International Joint Conf. on Artificial Intelligence (IJCAI-95)*, ftp://ftp.cs.ubc.ca/ftp/local/poole/papers/lprc.ps.gz, pp. 150–157.

Poole, D. (1996). A framework for decision-theoretic planning I: Combining the situation calculus, conditional plans, probability and utility, *in* E. Horvitz and F. Jensen (ed.), *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI-96)*, Portland Oregon, pp. 436–445.
*http://www.cs.ubc.ca/spider/poole/abstracts/iclsc.html

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty, *Artificial Intelligence special issue on multi-agent systems* **to appear**. http://www.cs.ubc.ca/spider/poole/abstracts/icl.html.

Poole, D., Mackworth, A. & Goebel, R. (1997). *Computational Intelligence: A Logical Approach*, Oxford University Press, New York.

Puterman, M. L. (1990). Markov decision processes, *in* D. P. Heyman and M. J. Sobel (ed.), *Handbooks in OR and MS, Vol. 2*, Elsevier Science Publishers B. V., chapter 8, pp. 331–434.

Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression, *in* V. Lifschitz (ed.), *Artificial Intelligence and the Mathematical Theory of Computation: Papers in Honor of John McCarthy*, Academic Press, San Diego, California, pp. 359–380.

Reiter, R. (1996). Natural actions, concurrency and continuous time in the situation calculus, *Proc. Fifth International Conf. on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, pp. ??–??

Savage, L. J. (1972). *The Foundation of Statistics*, 2nd edn, Dover, New York.

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

Von Neumann, J. & Morgenstern, O. (1953). *Theory of Games and Economic Behavior*, third edn, Princeton University Press, Princeton.