What's in a Decision?

Erik Sandewall Linköping University, Linköping, Sweden

Abstract

We address knowledge representation issues that arise when designing systems with cognitive autonomy and, in particular, the representational framework for deliberation about courses of action and for monitoring the execution of actions, as well as the momentary decisions that mark the transition from deliberation to action.

1 Topic

A cognitive robot must be able to work towards given or chosen goals by selecting plans, executing them, detecting plan failure, and replanning when needed. It must furthermore perform these functions in real time, and take into account that when the available time is limited, planning and replanning will consume some of it. It must take into account that actions in the real world may fail for a variety of reasons, and that both sensors and actuators are imprecise, while at the same time it must be able to use available knowledge about *how* imprecise and *how* unreliable those devices are. Finally, it must also be able to conduct a dialogue about its own deliberations, actions, and goal achievement.

In the mobile robot applications that we are working with, the difficulties in designing the cognitive level of the system are mostly due to the requirement of such an integrated and self-applied combination of perception, deliberation, and action, and not so much to the complexity of the plans as such. A precise and comprehensive model of cognitive autonomy is important for several reasons: as a basis for effective software design, and as a basis for theoretical characterization of the capabilities of a given system, and it is also a necessity for the reflexive situations where the system deliberates about, or communicates about its own performance.

In addressing the modelling, or knowledge representation of and for cognitive autonomy, the present paper emphasizes the broader, conceptual issues, at the expense of formal detail and details about the implemented system which have already been treated in more detail in our other publications.

The discrete step from deliberation to action is called a *decision*: in addressing a given goal, a cognitive robot will naturally deliberate for a while on the pros and cons of a few candidate courses of action, including nonaction as one of the options, and then choose one of them at some point. After that, the chosen plan becomes part of the robot's agenda, and the other ones are reduced to the status of might-have-beens. Finding an adequate representation for decisions in this sense is quite nontrivial.

2 Formal framework

We are using our X logic for reasoning about actions as the formal framework. Since the logic is not the main topic in this particular article, we shall only specify it in outline. In line with the anonymous reviewing principles of the ECAI conference we do not identify X in this manuscript. We make a brief introduction to it instead, with some superficial change of notation compared to previously published texts.

2.1 Formal language

X logic is a variety of the modern event calculus, where the values of properties are recorded at discrete or continuous points in time, and actions are considered to take place over intervals of time. The following notation is used:

$Holds(t, f, v)$: The fluent f has the value v at time t.

$Init(t, a)$: The agent initiates the action a at time t.

$Exec(t, x, a)$: The action a is executed from time t to time x.

$Xsucc(t, x, a)$: The action a is executed successfully from t to x.

$Xfail(t, x, a)$: The action a is executed from time t but fails at x.

$Xnorm(t, y, a)$: The action a is executed since time t, and is still performing normally at time y. (This can be said for each y during the execution of the action).

Fluents can be atomic or composite; the expression $age(john)$ is an example of a composite fluent which will have different values at different times. Actions can similarly be atomic or composite. The expression $goto(the-kitchen)$ is an example of a composite action. Actions can also be composed using composition operators such as $seq(a, b)$ for sequential composition and $if(f, v, a, b)$ expressing that if the fluent f has the value v then the action a is executed, otherwise the action b.

2.2 Ontology of Actions

The set of action-related predicates given above is actually somewhat redundant, and some of the predicates may be defined in terms of the others, but it is convenient to have all of them when writing axioms that characterize actions. The predicates $Init$ and $Exec$ are related by

$Init(t, a) \rightarrow \exists x[Exec(t, x, a)]$

We also have

$Xsucc(t, x, a) \rightarrow Exec(t, x, a)$,

$Xfail(t, x, a) \rightarrow Exec(t, x, a)$, and

$\neg(Xsucc(t, x, a) \wedge Xfail(t, x, a))$.

If both $Exec(t, x, a)$ and $Exec(t', x', a)$ then the intervals $[t, x]$ and $[t', x']$ can not overlap in more than one point. The full set of axioms for these predicates has been published before. The predicate $Xnorm$ is new for the present article, and will be explained in section 4.1.

We take the philosophical position that an action is executed when and as long as the agent chooses to execute it. The action may succeed or fail; the definition of when it shall be called a failure is the choice of him or her that designs the formalization. It is assumed, however, that actions 'usually' succeed, and that the robot chooses plans consisting of actions that will achieve current goals provided that the actions succeed. Plan execution therefore involves checking that the actions being performed come out as intended, and to modify the behavior when an action fails.

3 Deliberation using Natural Deduction

The characteristic behavior for the cognitive robots in our system is not to plan, but to maintain an agenda using prediction. At each point in time, the system has an agenda for what it is going to do next. It continuously performs several functions in parallel:

- Receiving percepts in the form of logical formulas
- Executing the next action(s) in the agenda, and identifying their success or failure based on incoming percepts
- Predicting the likely development in the world given the incoming observations and its own intended behavior in the agenda
- Modifying the agenda if the predictions indicate the consequence or the significant possibility of a negative development
- Also, modifying the agenda using planning or a plan library, in such a way that given or chosen goals will be achieved provided that actions in the agenda execute successfully.

The action execution item also includes the procedure for proceeding to the next action in a chosen plan (a plan is a composite action using composition functions such as seq and if) when an earlier action has been completed successfully.

The reason why prediction is so important is because of the complexity of the world and the many possibilities for spontaneous events in the world, concurrently with the execution of a proposed plan. This is not an issue in a simple, STRIPS-type world that conforms to the frame-problem assumptions, but in more realistic environments the cognitive system has to deliberate on what will be the effects of the plan in the particular, complex situation that it meets. In particular, it has to deliberate about what may possibly go wrong when a proposed plan is carried out.

It is important to distinguish between the *agenda* and the *option set*. The agenda specifies what the robot has committed itself to doing; the option set is a collection of hypothetical ways of carrying out agenda items, often with several options for the same agenda item. The choice of representations for options and for the agenda is therefore a key issue.

Each option may be associated with a more or less extensive set of predictions and other information. Even if the system has a library of plans for each type of goal, in every concrete situation it will have to check out one or more of the proposed plans by relating it to the situation at hand, and predict the near future if the plan is executed in the current context. Due to the potentially large size of the set of predictions for one option, it is not reasonable to represent them in one single formula.

Our approach to this problem is to use a slightly modified form of natural deduction, where the modification consists of (1) making it possible to refer to blocks and to associate information with them, (2) making it possible for variable symbols to change status and become constants, and (3) making it possible to remove the hypothetical character of the block, in a way that will be described here.

The following is how it works. Suppose the agent has a current goal g which is modelled as a boolean-valued fluent, and the plans $seq(a, b)$ and $seq(e, f)$ have been identified as options, e.g. by virtue of the following axioms:

$Xsucc(t, x, seq(a, b)) \rightarrow Holds(x, g, T)$

$Xsucc(t, x, seq(e, f)) \rightarrow Holds(x, g, T)$

The problem, of course, is that the agent is only able to *try* one or the other of these plans; it is not able to guarantee that the actions in the plan are successful. The plan may fail due to events that come up in the course of execution and that could not have been predicted, but some cases of action failure are predictable, and the purpose of the prediction machinery for the options is to identify action failure as often as possible in order to reduce the number of times the system fails and must retry.

The system represents each option as a block in the sense of natural deduction. Each block starts with a hypothesis concerning the choice of actions. In our example it would be, for one of the blocks, $Init(t, seq(a, b))$ saying that the plan or composite action $seq(a, b)$ is initiated at some time t. The choice of t is still to be determined, since the system needs to deliberate first. The actual value of t will be determined if and when the action is actually initiated, so for the purpose of deliberation t must be given as a variable. However, the hypothesis may include a restriction on when the action is to start, for example as follows:

$Init(t, seq(a, b)) \wedge t < 290$

In general, action hypotheses have the form $Init(t, \alpha) \wedge \Psi$ where α is an action expression, and Ψ is a formula only containing t and free variables in α as free variables.

The subsequent lines in the block contain conclusions that are obtained using the inference rules of natural deduction applied to formulas on earlier lines in the present block *and in all surrounding blocks.* In this way, the block starting with "initiate the plan $seq(a, b)$" will contain the predicted future in the current context and conditional on that action.

Intermediate consequences in that block shall have the form of predictions. A *fully deterministic prediction* has the form

$Xnorm(t, t+d, a) \wedge \Psi \rightarrow Holds(t+d, f, v)$

and specifies unambigously what feature-value will apply at exactly what time in the execution of the action, relative to the time it started. A *deterministic prediction* relaxes the determinism with respect to time, and has the form

$\exists d[d1 < d < d2 \wedge$
$(Xnorm(t, t+d, a) \wedge \Psi \rightarrow Holds(t+d, f, v))]$

Nondeterministic predictions are similar formulas that however allow a disjunction between several *Holds* literals in the consequent. They will not be considered in the present article; the main difficulty with them is on the computational side.

Restrictions on delays occur repeatedly in our formulas, and for convenience we introduce the following abbreviation:

$\Gamma d[d1, d2]\Phi \equiv \exists d[d1 < d < d2 \wedge \Phi]$

Predictions can be obtained in the following ways:

- By instantiation of *prediction axioms* that have the general form of predictions, but where universally quantified variables (or free variables with implicit universal quantification) may occur in a, f, and v.

- Using *causal laws* that may have the following form in many simple cases:

  $Holds(t, f, v) \wedge \Psi \rightarrow Holds(t+d, f', v')$

  or more generally

  $\Gamma d[d1, d2]Holds(t, f, v) \wedge \Psi \rightarrow Holds(t+d, f', v')$

  where the restriction Ψ is limited to free variables in t, f, and v like before, and where $d2$ is typically small or even zero.

It is clear that the combination of a prediction and a causal law under Modus Ponens results in another prediction, if the appropriate literals match. In this way the system is able to reason about side-effects or 'ramifications' of the plan that is being considered.

The predictions inside a block may conclude when a *goal prediction* has been obtained, that is, a proposition of the form

$\Gamma d[\lambda, \kappa]Xnorm(t, t+d, a) \rightarrow Holds(t+d, g, T)$

where g is the given goal, the terms λ and κ may contain free variables occurring in the action term a, and where in addition one has not been able to infer $\neg Xnorm(t, t+\kappa, a)$ in the same block.

Notice that in those cases where special circumstances predictably prevent the normal outcome of an action or sequence of actions, then either one has been able to infer $\neg Xnorm(t, t+\kappa, a)$, or one has not been able to infer the implication shown above.

4 Execution using Grounded N-D Blocks

4.1 Monitoring the Execution of Actions

The cognitive robot is considered to receive percepts from its sensory system in the form of ground literals for the predicate *Holds*. Axioms that characterize the normal execution of actions typically have the form of predictions, in a general case, as

$\Gamma d[d1, d2]Xnorm(t, t+d, a) \wedge \Psi \rightarrow Holds(t+d, f, v)$

Typically, there can be several such statements with different constants $d1$ and $d2$. Together with a default whereby the extent of the predicate *Xnorm* is minimized, this provides a way of assuming that the action proceeds successfully until incoming observations refute the expectations, at which point the default for *Xnorm* has to be abandoned for the reverse conclusion $\neg Xnorm(t, c, a)$ for a particular time c when the irregularity was detected. It is natural that the agent then decides to terminate execution of the action, which means that $\neg Xnorm(t, c, a) \rightarrow Xfail(t, c, a)$ is a plausible behavior rule for the system. (The actual rule is more complex).

4.2 The use of Grounded N-D Blocks

The predictions about a course of action that were made in the deliberation phase, should of course be used in order to monitor its execution, by relating them to the stream of percepts in the form of ground literals for *Holds*.

Suppose that after due deliberation, the cognitive robotics system has decided to initiate the action plan $seq(e, f)$ at time 394. Then the proposition $Init(394, seq(e, f))$ must also be added to its body of known facts. The deliberative predictions are located inside a block where the initial hypothesis is of the form $Init(t, seq(e, f))$ where t is a variable. Every prediction inside the block, for example

$Xnorm(t, t+14, seq(e, f)) \rightarrow Holds(t+14, f, v)$

for some f and v, must be replaced by, or effectively imply

$Xnorm(394, 408, seq(e, f)) \rightarrow Holds(408, f, v)$

This can be obtained in conventional natural deduction, but by a fairly elaborate process: for every such prediction, one has to obtain the following proposition outside the block:

$Init(t, seq(e, f)) \rightarrow$
$(Xnorm(t, t+14, a) \rightarrow Holds(t+14, f, v))$

using the inference rule "implication introduction", then obtain

$\forall t[Init(t, seq(e, f)) \rightarrow$
$(Xnorm(t, t+14, a) \rightarrow Holds(t+14, f, v))]$

using the inference rule "generalization", then instantiate t to 394, and finally use Modus Ponens.

The following technique eliminates the need for this obviously redundant process. The syntax is defined in such a way that one and the same symbol, for example $t29$, can be a variable at one point during the inference process, and a constant at a later time. The specification of which symbols are variables and which are constants is a part of the state of the modified natural-deduction process. The variables in block-forming hypotheses are chosen so that new variables are used in every new block that is introduced. When an action plan is effectuated and a proposition e.g. $Init(394, seq(e, f))$ is accepted, then any block starting with a generalization of that proposition e.g. $Init(t29, seq(e, f))$ changes status from being a normal natural-deduction block to being a *grounded* block, the symbol $t29$ changes status from variable to constant, and the formula $t29 = 394$ is also accepted (i.e., added to the body of believed facts). The grounding of the block means that all propositions in the block, including also its hypothesis, become ordinary propositions on the level outside the block.

If the block contains sub-blocks, then they retain of course their normal block status, but one level less deep. Grounding is only defined for top-level blocks, since we have not seen any need to perform it on lower-level blocks.

5 Deciding to Act

5.1 Ontological Status of the Decision to Act

We have described how the deliberation of possible courses of action is done inside natural-deduction blocks, and how the execution of a plan takes place on what is from a logical point of view the top level of natural deduction (not inside any blocks) although in a practical sense the statements inside grounded blocks are still administrated as part of the block. While the plan is being executed, the predictions in the grounded block(s) are retained together with a gradually extended set of percept propositions, whereby predictions can be related to observed facts. Both the deliberation and the execution phase can therefore be characterized in logicist terms, and with a machinery that differs only in limited and well-defined ways from standard natural deduction.

The transition from deliberation to plan execution is a non-trivial one, however. At some point in the course of deliberation, the cognitive robot has to make up its mind and choose one or the other of the options that are being considered, together with a starting-time and possibly other parameters for the activity. Can the decision to act also be characterized in logicist terms, or does it forever have to belong to the procedural and irrational domain?

In fact it is quite desirable to characterize it logically, since from a dialogue point of view the robot ought to be able to utter and to understand statements such as *at time t I decided to start doing action a at time u* or *the decision to do a was made too late* or even *now you must make up your mind about which car to take, otherwise it will be too late to get there in time.*

In our terms, therefore, it is natural to consider the step of adopting one of the options as an action in itself - a *decision action.*

In order to elaborate on decision actions, which is the central topic of this article, we must first of all resolve two relatively minor ontological issues. First, should the decision action be considered to have the form "start plan p now" or "start plan p at time d time units into the future from now" (with $d \geq 0$)? We choose the former alternative, since the latter case can always be represented as immediately starting a plan of the form $seq(wait(d), p)$

The second question is whether the decision action should best be considered as a momentary one or as an action with a duration in time, or maybe not as an action at all but as the value of a decision fluent. In our representation, we follow the pattern in natural language and view a decision, of the type "decide to", as an instantaneous action. However, it has some special and idiosyncratic logical properties, in particular with respect to its execution.

5.2 The Comparison of Options

The decision action has a peculiar extra-logical character. It marks the transition from the *deliberation about* a course of action to the *execution of* that course of action, and we have shown in outline how both the deliberation and the execution are being performed in logicist terms, and also how the technique of grounding natural-deduction blocks allows a close and direct connection between deliberation and execution. The decision-action itself is an enigma, however.

Consider the following example. Over a period of time, the cognitive robot deliberates on several options, each of which is a possible plan of action. It accumulates predictions for what is going to be future states of the world in each of the options, and in this way it may foresee good things and bad things that happen in each of those hypothetical futures. Making these predictions is an anytime computational process: the longer time one uses for it, the more predictions can be made, although at the same time if the predictions are nondeterministic then some of the future branches can also be invalidated by the stream of incoming percepts.

Then at some time the agent makes up its mind, which leads to the grounding of one of the options, and to abandoning the deliberation over the others. The simplest possible criterium for making the decision is if a goal prediction (defined above) has been obtained in one of the options, and the system considers that as sufficient reason to pick that option without further considering the others. But even this very simple policy can not be characterized in normal logical terms. One might try an implication of the form

$$(IS \wedge \Psi \rightarrow GP) \wedge \Psi \rightarrow IS$$

where $IS \wedge \Psi$ is the initial hypothesis in the option block, and GP is the goal prediction, as a way of saying that if goal prediction is achieved and the side-condition Ψ is satisfied, then decide on the plan in that option. However, then the logic would entail that all goal-achieving plans are in fact executed by the system, which is not what we wish to say.

The problem is, of course, that even this simple decision rule relies on a reference to the status of the deliberations on the *other* options, besides the one that is presently chosen. Also, this is in fact the poodle's kernel, since other and more complex decision criteria can also be accounted for, if only we are able to let one option refer to the deliberation status of the others at the current time.

Consider, for example, the following decision rule which is reasonably realistic, and which for simplicity we formulate for the case of choosing between two options only. (It generalizes easily to the case of more than two options). For each option and at each point in the deliberation, the system identifies four goodness values for the option in question, namely certain goodness (CG), possible goodness (PG), certain badness (CB), and possible badness (PB). The idea is that each option has its good and bad aspects, and that you can quantify them separately, and also that there is an interval of possible values for each of goodness and badness due to incomplete deliberation and intrinsic uncertainties about the outcome of actions in the world. We want the system to compare options based on their respective values for these four parameters, with system specific rules for weighing the good against the bad, and for the degree of optimism or pessimism. It is not difficult to write rules that calculate and update these parameters within each of the option blocks, and it is also easy to compare sets of these parameters if only one is able to make reference to the current status of deliberation in each of the blocks.

In general, therefore, we design the deliberation system so that there is a set of *option evaluation parameters*, and so that every option block assigns values to these at each point during the deliberation process. But how can the behavior rules refer to the values of those parameters?

The first problem is that the evaluation changes as the deliberation proceeds. This is resolved by characterizing the current state of deliberation in terms of the system's real time. The option evaluation parameters are therefore in principle no different from sensor values, which are reported to the higher levels of the system in the form of percepts.

There are two more problems: how can ND blocks be referenced, and how is the choice of an option effectively translated into action? We consider the questions in that order.

5.3 Representing References to N-D Blocks

Decision-making rules have to refer to the various option evaluation parameters, such as 'certain goodness' and 'possible badness' that were mentioned above by way of example. These are parameters that refer to the current consequences of a particular hypothesis formula, but it is possible that one and the same hypothesis is used to form a block several times during the life-length of the system. Therefore, and also for notational convenience, we introduce an option naming scheme. An additional type of *option names* is added to the other types that are already present in the X logic (timepoints, fluents, values, and actions, in particular). The predicate $Option(o)$ of one argument is also introduced, where the argument shall be of type option name. Instead of selecting propositions with $Init$ as block hypotheses, we now write *block definitions* such as

$$Option(o12) \rightarrow [Init(t, seq(a, b)) \wedge t < 290]$$

and then start the natural deduction block with $Option(o12)$ as the hypothesis. The block definition is obtained by application of a particular inference rule which says that formulas of this kind can be added at arbitrary times, provided that a new option name is used each time. Note that the block definition occurs on the top level of natural deduction, and not inside the block starting with $Option(o12)$. Note also that this amendment is harmless in the sense that it does not change the set of theorems not containing the new constructs.

Inside an option block, the values of option evaluation parameters are expressed using the predicate $Holds$, with the names of the evaluation parameters as fluent-valued functions. Deliberation can be used to identify the values for option evaluation parameters. For example, at time 255 the deliberation in option block $o12$ might assert $Holds(255, possBad(o12), 40)$ with 'possible badness' on a scale from 0 to 100. This estimate will be known to points outside the block, and to points inside sister blocks, using the formula

$$Option(o12) \rightarrow Holds(255, possBad(o12), 40)$$

If the evaluation changes after some additional deliberation, then that will be reported with another value for $Holds$. Note that the $Holds$-$possBad$ proposition is best made inside the block, since its derivation will depend on facts in that block.

5.4 The Double Helix Architecture

The third and final question is how decisions actually get to be done. This can only be answered by referring to the robot's architecture. We are using the *Double Helix Architecture* (DHA), which has been described in a separate article. It is a synchronous architecture for cognitive autonomy, where there are two concurrent processes, deliberation and execution, that exchange information with fixed periodicity. The times of information exchange are called *crosspoints*, and at each crosspoint the execution process sends observation-based percepts to the deliberation process, and the deliberation process emits requests to initiate the execution of actions. Those action requests are the result of asserting propositions of the form $Init(t, a)$ in the notation being used here.

Between crosspoints only deliberation and execution go on, but there is also a certain amount of computation at the crosspoints. This computation is rule-based, and is defined by a number of situation-action rules. It is used, in particular, for administrating the initiation of actions as well as the conclusion of actions with success or with failure, based on events in the incoming flow of percepts. Finally it is used for administrating the transition from one action to the next in a composite action formed using the operator *seq*.

It is possible to view the Double Helix Architecture as an instance of the familiar three-level architectures, but what is original about it is its synchronous character, with the insistence on a fixed-interval, periodic framework for the computation, and the relegation of the stimulus-response or "middle" layer to only be performed quasi-instantly at the crosspoints.

5.5 Decision-Making in the DHA

We are now ready to discuss the exact point where deliberation leads to action: this happens in the situation-action rules that are performed in each crosspoint. These S-A rules are triggered by percepts. Ordinary percepts represent observations that originate in the physical sensors in the execution process. The evaluation parameters that originate in the workings of the deliberation process, are also made available to the S-A rules.

In general, the S-A rules tend to operate on value assignments of the simple form object - feature - value. This pattern is applicable for evaluation parameters as well; from the point of view of the S-A rules each option is an object and each of its evaluation parameters is a feature.

Just like other S-A rules are able to instigate behavior patterns at moment's notice, similar to human reflex motions, so can the decision S-A rules trigger an action based on the fact that the deliberation has been pursued long enough. And just like the deliberative system is able to observe the robot's undeliberated reflex actions in retrospect, as in *I just stopped instantly because my front bumper hit something*, so it is able to observe *that* it has made a decision. By inspecting the deliberation state of the selected option, it can also obtain and report an explanation of the reasons for its decision.

It is possible to restrict the design so that the agent does not conceptualize having made the decisions. The agent is then able to decide, but not to reason about having decided. On the other hand, once the agent has the concept of making a decision, it is also possible to modify the procedure so that the agent is aware of making the decision at the time it makes it. By comparison, it is easy to find counterparts of both kinds of decisions among people, that is, both the case where deliberation is directly translated into action, and the case where deliberation on the course of action finishes with a brief (hopefully!) deliberation on whether one has deliberated enough. In the robot, the latter behavior requires the system to be able to intercept the urge to act, and to let the S-R rules interact with the deliberation thread before the action is released.

6 Related work

There are many formalisms for reasoning about actions and change, and many architectures for autonomous agents and cognitive robots. The integration of logic and architecture is less common, and when done is often for the purpose of planning, e.g. in the work on Golog (Levesque et al, 1997). Early work on cognitive robots was based on a plan-then-execute principle (Green, 1969), but the need to interleave planning and execution was also recognized. Three-level architectures evolved around 1990 with this in mind, but very often the upper-most, 'deliberative' layer appears to have been vacuous.

The work of Lyons and Hendriks (1992) is close to ours in spirit. Their system consists of a *reactor* and a separate *planner*, where the reactor is a structured collection of sensor and actor processes that run continuously, and the task of the planner is to modify the reactor so as to bring it in line with current goals, and in spite of its name it is supposed to do prediction as well as plain planning. The framework for prediction and planning is significantly less general than the X logic for actions and change that is used here, however.

The first concrete proposal for a system that can reason in general about actions and change and where deliberation occurs concurrently with execution appears to be due to Kowalski (1995). One difference between our work and his is that our double helix architecture accomodates the character of realistic control systems.

Maes (1989) describes action selection as an emergent property of the internal dynamics of the system, which is a different approach to action decisions. The major limitation of her approach seems to be that it becomes very difficult to analyze and to characterize the properties of the resulting systems.

Shanahan (1998) and Pirri and Finzi (1999) have made concrete and detailed proposals for a logicist treatment of perception, which problem has not been considered in this article.

7 Conclusion

The present article has given an introduction to its topic, but excludes some parts of it due to space constraints, e.g. the treatment of decisions where one chooses between "act now" and "deliberate more". One interesting result here is the interdependence between logicist account and system architecture. The former is often used as a basis for the latter, but in our case the architecture also provides the solutions for two traditional problems in the logic system: referencing the present state of deliberation, and taking the step from deliberation about an action to actually performing it.

8 References

Green, C. (1969): Applications of Theorem Proving to Problem Solving. Proc IJCAI 1969, pp. 219-240.

Kowalski, R. (1995): Using Meta-Logic to Reconcile Reactive with Rational Agents. In K.R. Apt and F. Turini (eds): Meta-Logics and Logic Programming, pp. 227-242. MIT Press, 1995.

Levesque, H, R. Reiter, Y. Lesperance, F. Lin and R.B. Scherl: GOLOG: A Logic Programming Language for Dynamic Domains. J of Logic Programming, vol. 31 (1997), pp 59-83.

Lyons, D.M. and A.J. Hendriks (1992): A practical approach to integrating reaction and deliberation. Proc of First Int'l Conf on A I Planning Systems (AIPS), pp. 153-162. Morgan Kaufmann, 1992.

Maes, P. (1989): The Dynamics of Action Selection. Proc IJCAI 1989, pp. 991-997.

Pirri, F. and A. Finzi (1999): An Approach to Perception in the Theory of Actions: Part I. ETAI, Vol. 3 (1999), pp. 19-61.

Shanahan, M. (1998): A Logical Account of the Common Sense Informatic Situation for a Mobile Robot. ETAI, Vol. 2 (1998), pp. 69-104.