Appears in Proceedings of the Eleventh Biennial Canadian Society for Computational Studies of Intelligence Conference (AI-96). To appear in a future volume of Lecture Notes in Computer Science (Springer-Verlag).

Constraint-Directed Improvisation

John Anderson and Mark Evans

Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2. Email: andersj@cs.umanitoba.ca or evans@cs.umanitoba.ca

Abstract. We present *Waffler*, a novel architecture that allows an agent to perform in complex, dynamic environments in a timely manner through improvisation. Improvisation involves using a routine method of accomplishing an activity as a guide to satisficing behaviour, adhering to that method as closely as the current situation permits for economic reasons, and exploring the background knowledge from which the routine has arisen to supplement the routine and move beyond it when necessary. Agents employing this approach can follow a routine in the face of uncertainty and variability, and can apply a routine in a situation with novel aspects, satisficing to the degree that time is available. This paper describes the Waffler architecture's basis in constraint-directed reasoning, it's knowledge structures and processing mechanisms, and an implementation in a simulated environment.

1. Introduction

Reactive approaches to planning are becoming increasingly dominant in the field, in light of the demands of real-time problem solving. While purely reactive approaches have proven useful in limited domains, attention is now turning to approaches that integrate reactive and deliberative behaviour, for several reasons. For the most part, knowledge in complex domains cannot be completely structured to the degree that purely reactive systems require. For example, approaches that completely disregard plan knowledge on the part of an agent (e.g. [1,6]) instead insist for the most part upon a complete mapping between all potential situations in an agent's environment and responses on the part of the agent. Compiling such a mapping is impossible in most real-world domains, and in those where it is conceivable, the size of a network implementing such a mapping makes it unworkable [10]. These approaches also assume that decisions can be based entirely on local information rather than long-term or global goals.

More pragmatically, purely reactive approaches run counterintuitive to what we observe in the course of most human activities. While activities such as working on an assembly line (where each potential problem has been experienced many times and there is a limited number of possible interactions) fit a purely reactive model nicely, such activities are exceptions rather than the norm. In virtually any human activity, there will be aspects or components of the activity with enough structure (experience on the part of the agent and physical structure in the environment [16,13]) to support a compiled collection of responses. During the course of an activity such as preparing a meal, for example, much is routine in a general sense despite the complexity of the environment: we can immediately recall a routine that has been put together over the course of many previous episodes of behaviour (compiled plan knowledge). Portions of this routine with which we are extremely familiar may be compiled to the point of a complete collection of reactions. Conversely, there will also be a large component of any complex activity that cannot hope to have this kind of structure: anywhere where

every possible contingency cannot be anticipated. This includes performing an activity with which we are not *completely* familiar, performing the activity in conjunction with others in the short- or long-term, or performing an activity in a different situation than usual [2].

In order to apply a routine effectively and flexibly in the face of greater variability than can be completely anticipated, we possess a vast collection of more general knowledge that allows us to integrate alternatives seamlessly with our routine. We divert from our routine when it makes sense to do so, and return to it without anything like the kind of effort known to be required (e.g. [11]) to alter a stored symbolic plan. We can also use our routine as a weaker guide in conjunction with background knowledge to cope in a satisficing manner with even greater degrees of variation. For example, one can shop reasonably successfully even when in a hurry and in a strange supermarket; can prepare a meal easily in a friend's kitchen; and can sharpen a pencil without a great deal of intellectual work even if no sharpener is available. We commonly call the methodology behind such efforts *improvisation*.

Improvisation as creating minor to extensive variations on a routine in a satisficing manner in real time occurs in the vast majority of human behaviour, from theatre and music to cooking, driving, and architecture [14]. Space prevents a detailed examination of improvisation in human activities here; however, see also [2]. In AI, the term has been used previously, most notably Agre's [1] definition of *continually redeciding what to do*. This use, however, leaves out much of what improvisation as described above embodies, most notably a basis upon which to improvise.

This basis is compiled plan knowledge - the routines we acquire through many episodes of previous experience. During the course of improvisation, this compiled plan knowledge represents a resource that is relied upon to reduce the intellectual effort that would normally be associated with the activity, in order to perform in a timely manner. We rely on this resource strongly in cases where the current situation follows our previous experience. In situations where our previous experience differs, we can use our routine as a weaker resource: following it as closely as possible for economic reasons, and improvising on the routine by examining the associated background knowledge to the degree the situation warrants, to obtain alternative actions in order to supplement portions that differ or are inappropriate. The more novelties there are in a given situation, the less directly the routine can be used, and the more search is required.

This process of improvising on a routine can also occur when one wants to reason beyond the routine aspects that normally constrain our reasoning: when one wants to do better than the routine, for example, or when one wants to come up with creative new solutions using the resources at hand as opposed to those commonly associated with the activity. Improvisation is a naturally satisficing process, allowing the agent to follow its routine and obtain immediate possibilities for action, or to devote as much time as is available or as the agent deems necessary for the task at hand to reason more deeply about alternatives for action. Computationally, improvisation has several requirements: control over the extent to which background information is explored, both with regard to the time spent on any one decision for action and the extent to which such exploration is deemed more valuable than the agent's routine response; dealing with limitations on the amount of background information that can be

considered at once, and organizing this information such that the most valuable information can be examined first; the integration of multiple goals; and the use of limited perception to recognize new opportunities in light of the agent's intended activities, to name a few.

We have developed an approach to timely reaction and deliberation for complex domains, based on this view of improvisation, that does not assume the degree of knowledge structuring and completeness required by purely reactive systems. Where extensive structuring is available, the approach takes advantage of it, and where it is unavailable the system improvises upon the structure that is present. This approach is embodied in an agent architecture known as *Waffler*, and allows an agent to react to its environment immediately (following both local and global goals), as well as to devote as much deliberative activity as time permits and knowledge of the situation deems warranted. This system employs constraint-directed reasoning mechanisms to represent the agent's knowledge of activity, to control the extent to which the agent improvises (and thus the degree of satisficing), and to control agent components (e.g. deliberation) that directly affect the agent's ability to perform in real time. The remainder of this paper reviews the constraint-directed knowledge representation mechanisms employed by Waffler agents, describes the organization and processing mechanisms of the architecture, and presents an implementation of the architecture in a simulated domain.

2. Intentions and Improvisation

In order to have access to routine responses as well as the ability to reason with the background knowledge from which those responses were compiled, both components must be coherently organized in an interconnected knowledge structure. A Waffler agent's compiled plan knowledge and background knowledge are incorporated into distributed constraint-based knowledge structures known as *intentions*. An intention consists partly of a general description of how to perform some particular activity (the agent's compiled plan knowledge). For example, the core of the intention shown in Figure 1 is a collection of steps that involve making tea, and constraints describing the relationship between them. Plan knowledge in intentions includes not only direct recommendations for actions such as these (which include cognitive activities such as adopting further intentions), but preferences, restrictions, and guidelines for resources and methods of performing actions, as well as more abstract preferences, such as those for methods of controlling computational processes within the agent itself.

An intention also contains links to the background information out of which the agent's compiled routine has evolved. This knowledge (concrete and abstract concepts and knowledge of activity settings) is organized as a highly-interconnected network, and individual concepts may be linked to the intention as a whole (Figure 1 illustrates general knowledge of the kitchen setting linked to this intention), or to specific portions of the intention (individual steps, resources, or constraints). Other background information will be connected indirectly to this intention and others through connections to concepts that are directly connected. Because this network of concepts represents the knowledge from which the agent's compiled plan has been derived, the more likely a particular concept is to be of use when following a particular intention, the closer it will be connected to it. Additional concepts not connected to

this intention may also supply alternatives for activity, through recollection mechanisms described in the next Section.

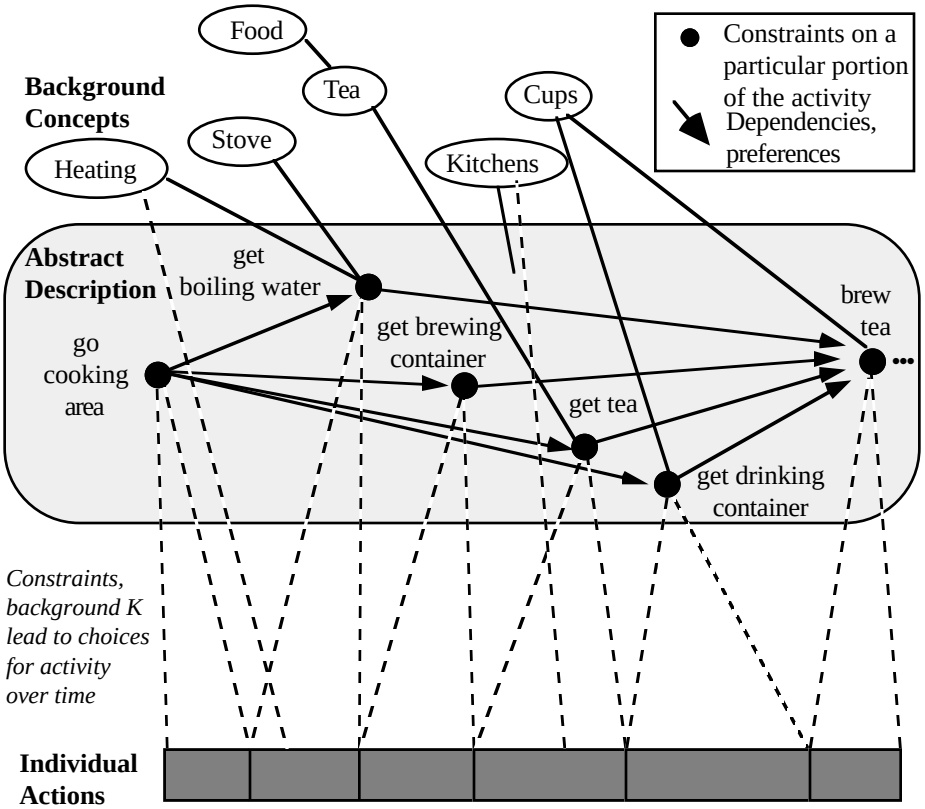

Figure 1. Example of an Intention.

This structuring allows the agent's routine to contribute alternatives for action and knowledge for decision-making immediately, and background information to do the same using a search process whose length will vary depending on how closely associated particular pieces of background knowledge are to the agent's routine. This in turn allows the agent access to immediate responses that are useful in the typical case of the activity, and the ability to search and deliberate as time and the significance of the situation permit. Intentions themselves are also linked hierarchically, allowing more general knowledge of activity to be accessed in the same fashion. The concept of intentions has been used previously, most notably by Bratman et al. [5]; however, intentions in the Waffler architecture are much more sophisticated than those used by Bratman. A Waffler agent's intentions are *active* guides to the agent's activity - they contribute alternatives for action and connections to background knowledge that can indirectly supply alternatives and constraints on behaviour, as opposed to influencing activity strictly through a sense of commitment to the intention itself.

The knowledge encompassed by an intention, including both the routine associated with an activity and the background knowledge that lies behind it, is represented largely through the use of constraints. Figure 1 illustrates only a few temporal and resource dependencies for the purposes of clarity. However, constraints may be applied to a broad range of concepts, from physical restrictions and requirements [9] to expectations of actions or other agents and control of agent components such as memory retention and deliberation [2,8]. In addition to these, constraints within this architecture are used to represent direct preferences for resources, actions, or activities; restrictions on agent focus (to particular tasks, knowledge or particular perceptual information); to represent agent policies for behaviour; and to represent normative responses to particular situations [2,4]. Constraints thus operate at multiple levels, from restrictions on objects and individual actions, to constraints associated with the agent's behaviour as a whole. High-level constraints in the latter group aid in selecting one action over another, limit processing at lower levels, and affect how constraints at lower levels are interpreted. Each level provides context to those beneath it, the same way the knowledge associated with one intention provides context for knowledge in intentions adopted in light of it.

As individual actions are selected (based on the constraints available), further intentions may be adopted. Actions may also be selected based on recommendations (constraints) from relevant background knowledge, as well as other intentions. A series of actions thus emerges over time as a result of the initial adoption of an intention in conjunction with others adopted at the same time and independent events that occur in the environment as the intention unfolds. This is illustrated in the lower portion of Figure 1.

The use of constraints as the primary knowledge representation mechanism directly supports the ability of an agent to perform in real time. For example, in an intention such as the one shown in Figure 1, the core routine may contain among other things a constraint indicating that the agent should prefer working with an electric kettle when boiling water as opposed to some other tool. This constraint expresses a preference that is normally applied in the course of the activity with no exploration as to the reasons behind the preference. We rely on such preferences heavily when we describe how we perform an activity, in order to limit consideration of the variability associated with such activities [2]. When this tool is unavailable or the agent wishes to reason beyond the routine (due to error, knowledge of potential error, or to high-level constraints such as *be-careful* that affect how an agent performs an activity), the agent can make use of further constraints behind that preference (background knowledge) that describe the role and function of the kettle in the overall routine. The agent can then use those constraints as a basis for reasoning about alternative ways of performing the activity, to the degree the agent wishes to devote intellectual effort to this. For example, constraints about heating water will lead the agent to a set of objects with characteristics suitable for this purpose.

Constraints external to an intention can also have immediate effects on it. The presence of a constraint such as hurrying (brought on by a combination of intentions or some external event) may have certain predictable effects that can be part of the routine itself. That is, the presence of a *hurry* constraint from outside the intention may allow certain routine components to become active that would be otherwise ignored. Such a constraint must also affect the agent itself: hurrying must affect how

much information the agent considers, strategies for deliberation, etc. The use of constraints in this manner will be described in the next Section.

There will clearly be a large number of constraints available in any significant domain. However, the agent's cognitive effort is for the most part not spent on looking for constraint violations, as in most constraint directed reasoning systems. While we are concerned about violations in some cases (e.g. expectations), here most constraints act positively: their presence compels the agent toward or away from specific courses of reasoning or activity, just as the landscape influences the direction of one's travel. The key to real-time performance is the selective processing of constraints in order to make satisficing decisions in the time available. This is done through the multi-level organization of constraints in tandem with the manner in which the Waffler architecture employs them.

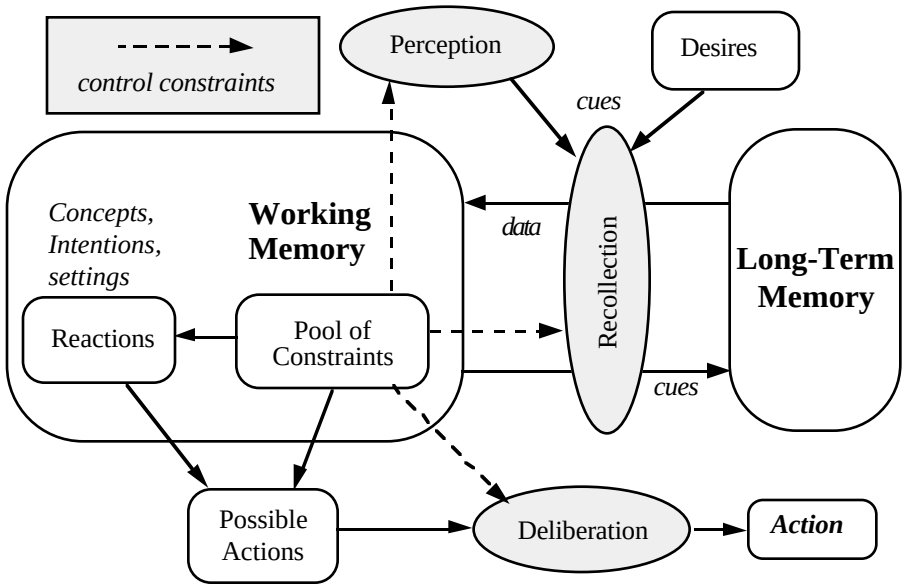

Figure 2. The Waffler Architecture.

3. The Waffler Architecture

The Waffler architecture consists of a set of parallel processes that adopt and apply intentions under resource and time constraints. An overview of this architecture is shown in Figure 2. The agent itself is divided into several computational processes and two major stores of knowledge. The agent's *long-term memory* contains all the possible intention and conceptual knowledge possessed by the agent. The agent's *working memory* is of a physically limited size and contains the active portions of the agent's knowledge (current intentions and a restricted amount of background knowledge). At any point in time, working memory will contain a number of intentions and concepts with attached constraints which reflect the agent's desires and perceived events in the environment thus far. These constraints will give rise to a particular set of alternatives for action, through methods that will be described shortly, and can also cause new items to be recalled from working memory (as can new desires

or perceptions), changing the milieu of constraints and thus the alternatives available. They will also affect the deliberative component which selects actions to be performed. The number of alternatives to be deliberated upon is directly proportional to the strength of the constraints in the agent's knowledge base, and will be small in cases where the agent is reasonably familiar with its situation (if the agent is unfamiliar, there will be additional possibilities and more reasoning required - the price to be paid for being a rationally bounded agent in a complex world). Deliberation may result in a choice for action, but may also involve waiting for more background information to be considered, for additional perceptual information, or for changes in internal state (and thus greater confidence in the agent's alternatives).

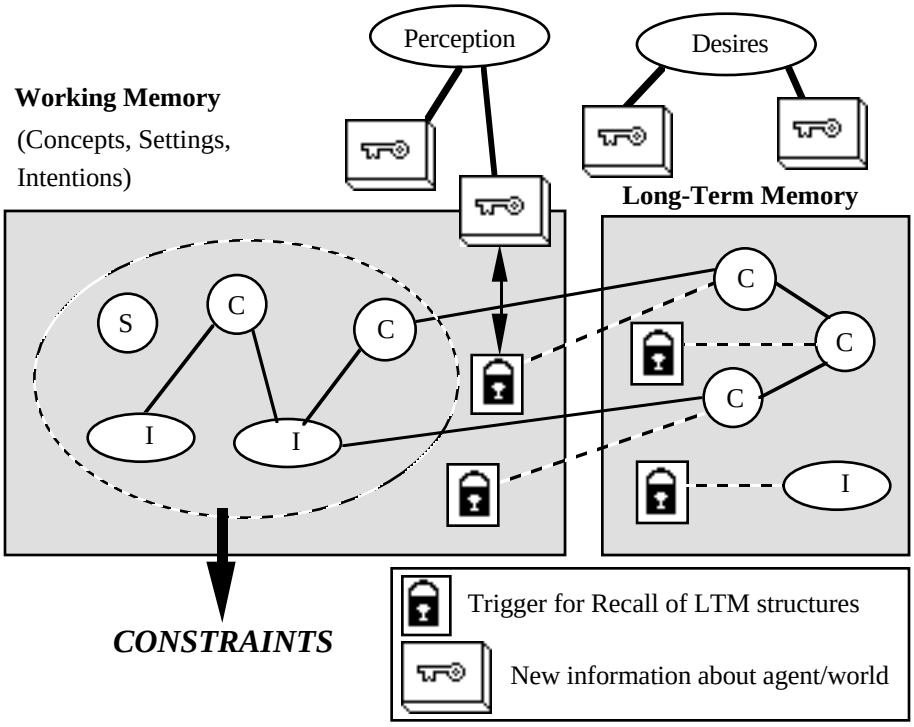

Figure 3. Details of Working Memory Recall.

The central role in this architecture is played by the agent's working memory, which directly supports the selective recall and processing of constraint knowledge. Working memory is of limited size, and represents the amount of information the agent can process in parallel (in our implementation a timeshared simulator is used, and the size of working memory represents the amount of information the agent can process in a time-slice). Any constraint in working memory is thus viewed as having immediate effects, and relations among concepts in working memory are assumed to be immediately realizable (via direct memory connections). These are not unrealistic assumptions, since the number of concepts or intentions allowed in working memory at the same time can always be set to a small enough limit to make this so. Items migrate from long-term to working memory through the use of memory triggers. Each object, concept, or intention may have trigger conditions associated with it that,

when satisfied, allow this item to be brought into working memory. These conditions may be satisfied by perceptual information, desires, or items already in working memory (Figure 3). Recollection is largely another responsibility of working memory (though external desires can also match concepts and cause their recall), and only trigger conditions currently in working memory are considered. When an item is brought into working memory, the triggers of objects connected to it follow, allowing them to be brought in if conditions permit. This process is inspired by that used by Pauker et al. [17] to deal with the large volume of knowledge in internal medicine, and directly supports the kind of background reasoning necessary in improvisation: beginning with the intention itself and gradually moving to concepts further associated with the intention as necessary.

As items are placed in working memory, others must be removed. The most appropriate memory management policy varies with the situation (e.g. if the agent is in a time-constrained situation with regard to some task, working memory should be biased toward concepts contributing to this task). This is also handled through constraints: constraints associated with intentions may suggest an appropriate policy, or more likely, higher level constraints (associated with the setting or more general background knowledge of activity) will recognize when specific policies are required. This illustrates the most significant aspect of the Waffler architecture: not only do constraints represent the structure of an agent's activities, they also control the architecture itself. Constraints are used to define a perceptual focus for the agent, and can also focus the agent toward retrieving particular concepts from long-term memory and thus follow a particular line of reasoning. As will shortly be illustrated, constraints can also directly affect deliberation, limiting the effort put into any particular decision or toward any particular activity. The uses of constraints within this architecture are summarized in Figure 4.

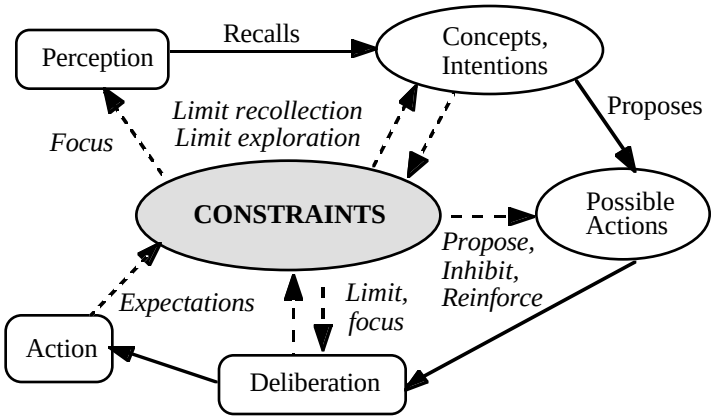

Figure 4. Constraints in the Waffler Architecture.

As already mentioned, the more constraints available to working memory, the fewer alternatives need be deliberated upon, and hence deliberation is secondary to working memory in this approach. However, because working memory is very limited, the agent must often deliberate with an incomplete picture of its overall choices, and thus deliberation is still significant. Individual constraints in working memory influence the agent toward (or away from) some alternative for action through a quantitative or

qualitative measure of utility. This measure is composed of an innate estimate of the constraint's importance, modified by the importance of the chain of intentions through which the constraint has come into working memory and by specific conditions the constraint is concerned with. Utility is thus situation- and activity-dependent.

Each constraint advocating or refuting some alternative to action contributes its utility toward (or away from) that alternative. At any time, a given number of constraints (those that have passed through working memory) will have contributed their utility, and a given number (those attached to relevant concepts not in working memory) will have not have contributed. In order to make decisions using incomplete information such as this, the agent has in its working memory a constraint on utility, representing the minimum utility necessary for an alternative to be acted upon. This constraint may be global or local to a particular activity the agent is performing, and serves to limit deliberation by allowing the agent to select the most important alternatives. It is not static, and can be affected by intentions and concepts in working memory as well as higher-level knowledge. For example, when conflicting intentions are present, high-level knowledge may alter the agent's utility constraint to be higher in order that the agent have greater confidence that it is selecting the right action in these situations. If no action can be performed, the agent may wait for more information to be processed through working memory. This affords the possibility of a higher utility for one or more alternatives, or for constraints in working memory to alter the agent's utility constraint, allowing less highly-ranked alternatives to be selected. It also presents the possibility that the agent could miss some time-constrained opportunity.

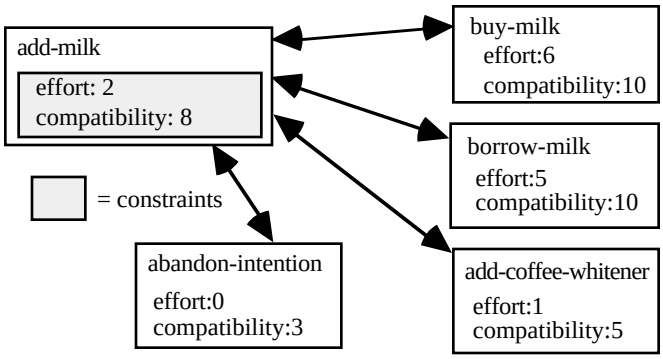

Figure 5. Deliberation using Effort and Compatibility.

Utility may also be split up into separate factors, each of which can be weighted. For example, Figure 5 illustrates an example wherein utility has been split into measures of effort (the amount of effort associated with the alternative) and compatibility (how compatible it is with the agent's routine response). Along with timeliness, we have found these to be the most important factors in the utility of any potential action in the course of human everyday activities [2]. In this example, the agent needs to add milk to a dish it is cooking, and has realized it has none. Several alternatives arise out of the constraints involved, each of which has particular estimates of effort and utility (shown as quantitative measures, but not restricted to be so) put forward by constraints in the agent's working memory (e.g. buying milk may be affected by the constraint that a store must be open, which affects the effort of this alternative). None of these choices is suitable for these constraints, and so the agent has no action to perform

(unless other intentions bring about additional possibilities having nothing to do with this). The agent may wait for additional information to alter these measurements, or knowledge associated with this activity or with activity in general may alter these thresholds so that one or more actions is possible. Waiting may also affect the importance of the intention that brought about this activity (or intentions further back in abstraction), which may further alter these constraints.

4. Implementation and Summary

We have implemented this architecture using the Gensim simulation testbed [3] under Macintosh Common LISP to provide a simulated world for the agent to inhabit. Gensim manages an object-oriented representation of an environment (including the abilities of agents), presents object-attribute descriptions of objects for agent perception (using a model of the agent's limited sensory abilities), and handles change in the environment through descriptions of the agent's activities in the previous time interval and its own model of the agent's abilities and the rest of the environment. Gensim manages agents with parallel processes through timesharing, and provides realistic simulations by enforcing a strict separation of agent and simulator knowledge. When agents wish to reference domain objects in descriptions of their activities or for specifying perceptual focus, for example, they must do so by describing the object relative to others (a deictic [1] approach) or by referencing their own internal symbols designating that object - they cannot access the simulator's knowledge to tell it what they are referring to.

The current environment is a simulated kitchen, in which accidents and other unexpected timely events can occur in the midst of the agent's ongoing activities: tasks such as making tea, answering the telephone, cleaning up, and other activities that normally take place in a kitchen. The agent can make use of tools such as electric kettles, spoons, etc., that are stored in cupboards and drawers. Agent intentions can be implemented as a constrained collection of steps, or as a function that can be called to generate a recommendation. The extent of the knowledge required for these activities necessitates that the implementation of the agent's intentions and background knowledge be incomplete. Portions of intentions and the background knowledge behind them are implemented in detail (e.g. the mechanics and background of acquiring boiling water) as opposed to having more extensive intentions in less detail.

The agent inhabiting this environment consists of five processes implementing the components shown in Figure 2. The first of these processes takes object descriptions (attributes and values) provided by the simulator as perception, and integrates these with objects in the agent's world model. Upon seeing a cup on the ground, for example, the agent must reason as to whether this is the same cup that was supposed to be in its hand or some newly discovered object. Perceived objects or events are also matched to all the triggers in working memory, which may in turn recall new concepts. The second process simulates the parallelism inherent in the agent's working memory: it polls each of the intention instances in working memory for its recommendations and attempts to activate all of the constraints in working memory. If an intention consists of a group of steps, any step not blocked by constraints is recommended; if the intention consists of a generator function, the function is called to produce a recommendation. Each intention step has a default utility, which is modified by the utility of the overall chain of intentions. This process also processes each

object in working memory, checking for constraints that may make direct recommendations or indirectly alter working memory. Links between objects in working memory are also processed. If the agent is looking for an object to fill a specific role, for example, and that role and object explicitly stated to be able to fill that role are in working memory, the connection is immediately made.

The third process implements deliberation by finding the most appropriate of the recommended actions by selecting that with the highest utility. It is possible to define constraints on minimal utility levels as described in Section 3, and so it is entirely possible to have no action reach a sufficient utility to be viewed as a suitable action. In such a situation deliberation can simply ensue over a number of cycles. Additional evidence may accumulate and allow some alternative to move beyond the stipulated limit, or some (internal or external) event may cause the utility constraint to be altered or abandoned. Following deliberation, another process is used to specify a focus for sensory information from the simulator and to commit to the agent's chosen course of action (if any) by communicating both of these items to the simulator.

The final process is not part of the overall architecture, and is used to simulate the need for working memory management that would accompany much larger domains. The current size of the domain (approximately 25 classes of physical objects and abstract concepts) does not adequately overload working memory to the extent that a real kitchen would, and rather than accepting the behavioural bias that would come from restricting working memory to a ridiculously small size or simply allowing everything in working memory at once, we instead remove unused components from working memory on each cycle. Items in working memory are timestamped, and if a concept or intention has not been referenced during the current cycle or the previous one, it is "forgotten" by removing the item from working memory (but not long-term memory).

We have used this implementation in several detailed examples [2] illustrating its ability to cope with unexpected events (errors, adapting to missing tools, competing intentions) during the course of an agent's activities. The largest of these examples begins with the agent adding various condiments to its tea, then recognizing a mess and adopting an attention to put away the objects used in its earlier activity (as a result of an active constraint to keep the kitchen clean). The telephone rings, and the agent recognizes a time-constrained opportunity, raising constraints on utility and effort and causing it to immediately select the available option with a high utility - to answer the phone. The agent forms an intention to travel to the phone, and a simulator-implemented delay keeps the agent "talking on the phone" long enough to forget its original intention (to clean up the mess). When the call is complete, the agent has no intentions in working memory, gradually lowers the thresholds described in Section 3 and effectively spends time looking for something to do. Its original overall activity (making tea) is recalled first because of a higher priority, and the agent returns to its original locale, recognizes the mess once again, and resumes its cleaning activities.

Additional implemented examples demonstrate the flexibility of improvisation in allowing an agent to apply its routine in spite of novelties in the environment. In a situation where an agent is used to using an electric kettle to boil water, and this is made unavailable, the agent initially takes no action and instead begins to recall background information. Once enough constraints have been examined to make an alternative plausible (e.g. looking for a stovetop kettle will occur as an alternative

almost immediately, and even if this knowledge is not strongly rated the importance of having tea will eventually lower the agent's threshold on effort and compatibility to the level at which this alternative will be considered. As more time is available or the agent is given additional incentive for further exploration (e.g. the stovetop kettle is removed from the environment) the agent can reason about increasingly novel methods of improvising on its routine. In the same situation, if knowledge that an electric kettle is available somewhere is supplied, the agent will begin looking in plausible places (using its background knowledge of a kitchen) for its usual substitute even if a less-suitable tool such as a pot is immediately available. Eventually, alternatives such as those described above will become more attractive as looking for the original tool appears futile. Alternatively, the attractiveness of the electric kettle in its routine can be modified so that it appears immediately easier to use a pot to boil water than to bother looking for its usual choice. Transcripts of these examples and many more implementational details may be found in [2].

This implementation is limited in several ways. As described above, it is limited to partial intentions because of the extent of the common-sense knowledge required to perform an everyday activity such as making tea. In future, such commonsense domain knowledge will become more widely available as projects such as CYC [15] become mature. In the meantime, we will be experimenting with more complex domains within current limitations. More practically, this implementation is limited in that it simulates the parallelism inherent in working memory (though limits it through mechanisms described above), and in the fact that parallel actions are not considered. Cognitive and physical actions are also treated similarly (for greater realism, so that cognitive actions will take up time and affect the agent's response time). However, a lack of parallel actions means that an agent cannot perform a cognitive action (e.g. adopting an intention) and a physical action at the same time.

At a surface level, this architecture is most immediately compared to that of Bratman et al [5] in its use of intentions and in the ability to derive new options and deliberate about them. In addition to the differences in the concept of intentions described earlier, there are additional differences. This approach provides specific methods for generating and dealing with options as a core of the architecture itself (and has a means of controlling option generation), rather than leaving these and other components as implementation-dependent details. Constraints as a knowledge representation mechanism also allow the agent to reason as deeply or shallowly about a particular situation as time allows (and necessity demands), and support more sophisticated reasoning in terms of applying routines in novel situations. This architecture is also comparable to recent work in partial universal planning [7]. Dean's approach allows a partial universal plan to be employed by halting and extending the situation-action mapping when an unknown situation develops. This is useful only when the agent can afford to stop and plan (Dean assumes that only minor extensions will be required, and that a deadline for plan completion is known), and does not capture the ability to do the best the agent can given partially-compiled knowledge, general background knowledge, and a limited time to explore that this architecture embodies. Finally, this work may also be compared with the case-based activity work of Hammond et al. [12], which allows goals in a routine to be blocked, suspended, and later recalled. Through the application of constraints, this approach also supports this ability (when constraints do not block an activity, it can resume), but also supports the ability to perform in a satisficing fashion in real time.

While the major contributions of this architecture lie in extending mechanisms for real-time deliberation in complex domains, the development of this approach also makes major contributions to extending constraint-directed reasoning and in understanding human cognition in everyday activities. We are currently applying improvisation as a mode of interaction in multi-agent scenarios, examining practical reasoning about function during the course of improvisation, experimenting with various forms of memory management, and exploring the use of improvisation in tracking and predicting the behaviour of other agents and in intelligent human-computer interaction.

References

1. Agre, Philip E., *The Dynamic Structure of Everyday Life*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, 1988. 282 pp.
2. Anderson, John, *Constraint-Directed Improvisation for Everyday Activities*, Ph.D. Dissertation, Department of Computer Science, University of Manitoba, 1995. 389 pp.
3. Anderson, John, and Mark Evans, "A Generic Simulation System for Intelligent Agent Designs", *Applied AI* 9:5, 1995, pp. 527-562.
4. Anderson, John, and Mark Evans, "Constraints as a Basis for Real-Time Planning", submitted to the *Second International Workshop on Constraint-Based Reasoning*, Key West FL, May, 1996.
5. Bratman, Michael, David Israel, and Martha Pollack, *Plans and Resource-Bounded Practical Reasoning*, Technical Note 425R, SRI International, 1988. 28 pp.
6. Chapman, David, *Vision, Instruction, and Action*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, 1990, 244 pp.
7. Dean, Thomas, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson, "Planning with Deadlines in Stochastic Domains", *AAAI-93*, Washington, DC, 1993, pp. 574-579.
8. Evans, Mark, John Anderson, and Geoff Crysdale, "Achieving Flexible Autonomy in Multi-Agent Systems using Constraints", *Applied Artificial Intelligence* 6:1, 1992, pp. 103-126.
9. Fox, Mark S., *Constraint-Directed Search*, Ph.D. Dissertation, School of Computer Science, Carnegie-Mellon University, 1983. 184 pp.
10. Ginsberg, Matthew, "Universal Planning: An (Almost) Universally Bad Idea", *AI Magazine* 10:4, 1989, pp. 40-44.
11. Hammond, Kristian, *Case-Based Planning* (Boston: Academic Pr.), 1989. 277 pp.
12. Hammond, Kristian, Tim Converse, and Charles Martin, "Integrating Planning and Acting in a Case-Based Framework", *AAAI-90*, Boston, 1990, pp. 292-297.
13. Hammond, Kristian, and Tim Converse, "Stabilizing Environments to Facilitate Planning and Activity", *AAAI-91*, Anaheim, 1991, pp. 787-793.
14. Jencks, Charles, and Nathan Silver, *Adhocism: The Case for Improvisation* (New York: Doubleday), 1972. 216 pp.
15. Lenat, Doug, M. Prakash, and M. Shepherd, "CYC: Using Common-Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks", *AI Magazine* 6:4 , 1986 pp. 65-85.
16. Norman, Donald A., *The Psychology of Everyday Things* (New York: Basic Books), 1988. 257 pp.
17. Pauker, Stephen, G. Anthony Gorry, Jerome Kassirer, and William Schwartz, "Towards the Simulation of Clinical Cognition", *American Journal of Medicine* 60, 1976, pp. 981-996.