Action selection for Intelligent Systems

Cyril Brom (Charles University, Prague, Czech Republic)

Joanna Bryson (University of Bath, Bath, United Kingdom)

Action selection is a way of characterizing the most basic problem of intelligent systems: *what to do next*. In artificial intelligence and computational cognitive science, **the action selection problem** is typically associated with intelligent agents and animats – artificial systems that exhibit complex behaviour in an agent environment. The term is also sometimes used in ethology or animal behaviour.

A basic problem for understanding action selection is determining the level of abstraction used for specifying an 'act'. At the most basic level of abstraction, an atomic act could be anything from contracting a muscle cell to provoking a war. Typically for an artificial action-selection mechanism, the set of possible actions is predefined and fixed. However, in nature agents are able to control action at a variety of levels of abstraction, and the acquisition of skills or expertise can also be viewed as the acquisition of new action selection primitives.

Action selection could also be seen as the intelligent-agent version of the engineering discipline of systems integration. A good action selection mechanism allows a developer to decompose the problem of building intelligent systems into relatively simple components or *modules* and then reintegrate or coordinate the overall behaviour.

Most researchers studying AI action selection place high demands on their agents:

- The acting agent typically must select its action in dynamic and unpredictable environments.
- The agents typically act in real time; therefore they must make decisions in a timely fashion.
- The agents are normally created to perform several different tasks. These tasks may conflict for resource allocation (e.g. can the agent put out a fire and deliver a cup of coffee at the same time?).
- The environment the agents operate in may include humans, who may make things more difficult for the agent (either intentionally or by attempting to assist).
- The agents are often intended to model humans and/or other animals. Animal behaviour is quite complicated and not yet fully understood.

For these reasons action selection is not trivial – it both requires and attracts a good deal of research. There are two journal special issues coming out on the topic of action selection in the next 12 months: in the Proceedings of the Royal Society B there will be an issue on *Modelling Natural Action Selection*, and in Adaptive Behaviour there will be an issue on *Mechanisms of Action Selection*.

This paper briefly describes various practical approaches to action selection used in artificial systems.

Action selection mechanisms

The main problem for action selection is combinatorial complexity. Since all computation takes both time and space (in memory), agents cannot possibly consider every option available to them at every instant in time. Consequently, they must be biased, and constrain their search in some way. For AI, the question of action selection is: *what is the best way to constrain this search*? For biology and ethology, the question is: *how do various types of animals constrain their search*? *Do all animals use the same approaches*? *Why do they use the ones they do*?

One fundamental question about action selection is whether it is really a problem at all for an agent, or whether it is just a description of an emergent property of an intelligent agent's behaviour. However, the history of intelligent systems, both artificial (Bryson, 2000) and biological (Prescott, 2007) indicate that building an intelligent system requires some mechanism for action selection. This mechanism may be highly distributed (as in the case of distributed organisms such as social insect colonies or slime moulds) or it may be one or more special-purpose modules.

The action selection mechanism (ASM) determines not only the agent's actions in terms of impact on the world, but also directs its perceptual attention, and updates its memory. These egocentric sorts of actions may in turn result in modifying the agent's basic behavioural capacities, particularly in that updating memory implies some form of learning is possible. Ideally, action selection itself should also be able to learn and adapt, but there are many problems of combinatorial complexity and computational tractability that may require restricting the search space for learning.

In AI, an ASM is also sometimes either referred to as an *agent architecture* or thought of as a substantial part of one.

General approaches to artificial action selection

Generally, artificial action selection mechanisms can be divided into several categories: *symbol-based systems* (sometimes known as *classical planning*), *distributed solutions*, and *reactive* or *dynamic planning*. Some approaches do not fall neatly into any one of these categories.

Symbolic approaches

Early in the history of artificial intelligence, it was assumed that the best way for an agent to choose what to do next would be to *compute a provably optimal plan*, and then *execute* that plan. This lead to the *physical symbol system hypothesis*, that a mechanism that can physically manipulate symbols is both necessary and sufficient for intelligence. Many software agents still use this approach for action selection. It normally requires describing all sensor readings, the world, all of one's potential actions and all of one's goals in some form of predicate logic. Critics of this approach complain that it is too slow real-time planning and that, despite the proofs, it is still unlikely to produce optimal plans because reducing descriptions of reality to logic is a process prone to errors.

A classical "good old fashion AI" symbolic architecture is Soar (Soar project, 2006). It has been being developed and extended from about middle of the 1980s originally at Carnegie Mellon University and more recently at University of Michigan. It is based on condition–

action rules (see below). This architecture has recently become a powerful programming toolkit, which can be used for building both reactive/dynamic agents and classical planning agents, or a compromise at will between these two extremes.

Distributed approaches

In contrast to the symbolic approach, *distributed systems* of action selection actually have no one 'box' in the agent which decides the next action. At least in their idealized form, distributed systems have many modules running in parallel and determining the best action based on local expertise. In these idealized systems, overall coherence is expected to *emerge* somehow, possibly through careful design of the interacting components, or through some sort of reinforcement-based learning of inhibitory connections between modules. This approach is often inspired by *neural networks research*. In practice, there is almost always some centralised system determining which module is "the most active" or has the most salience. There is evidence real brains also have such executive decision systems which evaluate which of competing systems deserves the most attention, or more properly, has its desired actions disinhibited (Prescott, 2007). A classicall example of a distributed approach is an architecture of creatures from the computer game Creatures (Grand et al., 1997).

Dynamic (reactive) planning

Because purely distributed systems are difficult to construct, many researchers have turned to using explicit *hard-coded plans* to determine the priorities of their system (Bryson, 2000).

Dynamic or reactive planning methods *compute just one next action* in every instant based on the current context and pre-scripted plans. Essentially, they compute the following function:

$$S \times P \rightarrow A$$

Here, S is the set of all possible internal states (including memory), P is the set of all possible actual percepts, and A is the set of all possible actions.

A *purely reactive* system is one that requires no memory. The term *reactive planning* has been used since at least 1988 to mean a system that performs action selection with minimal (though seldom no) internal state, but rather doing simple look-up at every instant for the best action to perform right then. The term *reactive* has unfortunately now become a pejorative used as an antonym for *proactive*. Since nearly all agents using reactive planning *are* proactive (and may exploit memory as well), some researchers have begun referring to reactive planning as *dynamic planning*.

In contrast to classical planning methods, dynamic approaches do not suffer combinatorial explosion, at least not for the agent while it is performing action selection. The search problem is moved instead to the developers or learning system that creates the agent's intelligence. Reactive systems are sometimes seen as too rigid or narrow to be considered strong AI, since the plans are coded in advance. At the same time, natural intelligence can be rigid in some contexts although it is fluid and able to adapt in others (Gallistel et al., 1991). Sometimes the reactive approach is combined with classical planning, leading to so-called *hybrid approach* or *layered architectures* (Bryson, 2000).

Dynamic planning techniques are extremely popular in real-time interactive commercial applications, because they cope well with dynamic and unpredictable environments and

require limited CPU. In particular, they are used in *computer games*, *agent-based modelling, home/entertainment robotics* and even *the movie industry*.

Dynamic planning approaches

There are several approaches to dynamic planning, which we describe in this section.

Deterministic condition-action rules

One branch of techniques relies on *condition–action rules*, which come originally from the domain of expert systems. A condition-action rule (also known as an if-then rule or a *production*), is a rule in the form:

```
if condition then action
```

The meaning of the rule is obvious: if the condition holds, perform the action. The action can be either external (i.e., to operate on the environment), or internal (e.g., write a fact into the internal memory, or evaluate a new set of rules). The conditions are typically boolean and the action either performed or not.

The rules are organised in flat structures, as in the case of simplified subsumption architecture (Wooldridge, 2002), or in hierarchical structures, for example decision trees, which is the case of most other architectures, e.g. (Brom et. al, 2006, Bryson, 2001). Flat structures allows only for the description of simple behaviour, or else their complexity / hierarchical structure can be masked by a very intricate set of conditions.

The important part of the action-selection algorithms is a *conflict resolution mechanism*. This is a mechanism for solving a conflict among rules when more than one of their conditions holds in a given instant. The conflict can be solved for example by assigning fixed priorities to the rules in advance (as e.g. in Bryson, 2001), by assigning preferences (e.g. in Soar) or by exploiting a form of classical planning. Assuming a flat structure but with priorities as a method for conflict resolution, we can have a following simple plan for a robot that "picks up mushrooms":

```
1. if see_obstacle then change_direction
2. if basketful_of_mushrooms and picking is true then
   picking is false
3. if see_mushroom and picking then pick_up_the_mushroom
4. if noon and picking is true then picking is false
5. if home then put_down_basket and END
6. if picking is true then move_random
7. if picking is false then move_home
```

Fig. 1: A flat basic reactive plan (see Bryson and Stein, 2000), which uses condition-action rules with priorities. The robot will pick up mushrooms till 12 p.m., but must start in *picking* state and not at home. The example is taken from Brom (2006).

Such a plan can be made hierarchical through the obvious mechanism – by allowing the 'actions' of the rules to themselves be such plans or sequences.

Deterministic Finite State Machines

The *finite state machine* (FSM) is another type of model for system behaviour. FSMs are used widely in computer science and modelling behaviour of agents is only one of their possible applications. A typical FSM, when used for describing behaviour of an agent, consists of a set of *states* and *transitions* between these states. The transitions are actually condition–action rules. In every instant, just one state of the FSM is active, and its transitions are evaluated. If a transition holds, it activates another state. That means, that transitions are the rules in the following form:

```
if current-state and condition then activate-new-state
```

There are several ways to produce behaviour by an FSM. They depend on what is associated with the states and transistions by a designer – they can be either acts or scripts, and they can be associated with either the transition or with being in the state. An act is an atomic action that should be performed by the agent either on transition or continuously while the FSM is in the given state. A script describes a sequence of actions that the agent has to perform if its FSM is in a given state. If a transition activates a new state, the former script is simply interrupted, and the new one is started.

As can be recognized by the condition-action structure described above, an FSM is in some ways similar to a structured dynamic plan. However, FSMs require enumerating all possible states for the agent. When these are many, a hierarchical dynamic plan may provide a useful shorthand that is easier to maintain. On the other hand, FSMs are well understood formal structures. Bryson (2003) describes circumstances in which each representation might be preferable.

If an FSM script is complicated, it can also be broken down into several scripts, and a hierarchical FSM (HFSM) can be exploited. In such an automaton, every state can contain substates. Only the states at the lowest, ‘atomic’ level are associated with a script (which should not be complicated) or an atomic action.

Computationally, HFSMs are equivalent to FSMs. That means that each HFSM can be converted to a classical FSM. However, the hierarchical approach facilitates design since HFSMs are both easier to understand and to specify conditions for. Isla (2006) and van Waveren (2001) give examples of ASMs for computer game bots which use hierarchical FSMs. For a movie example, see Softimage/Behavior (2006).

Less discrete approaches

Both action-condition rules and FSMs can be combined with either stochastic preconditions or fuzzy logic. In this case, the conditions, states and actions are no longer boolean or deterministic; rather, they become probabalistic. Consequently, resulting behaviour can be smoother, especially in the case of transitions between two tasks. However, evaluation of the fuzzy conditions is much slower then evaluation of their discrete counterparts. Both the fuzzy and the probabilistic approaches are well described by Champandard (2003).

Figure 2 shows an overall architecture of a typical reactive agent, whose ASM is based on rules or FSMs, either crisp or fuzzy.

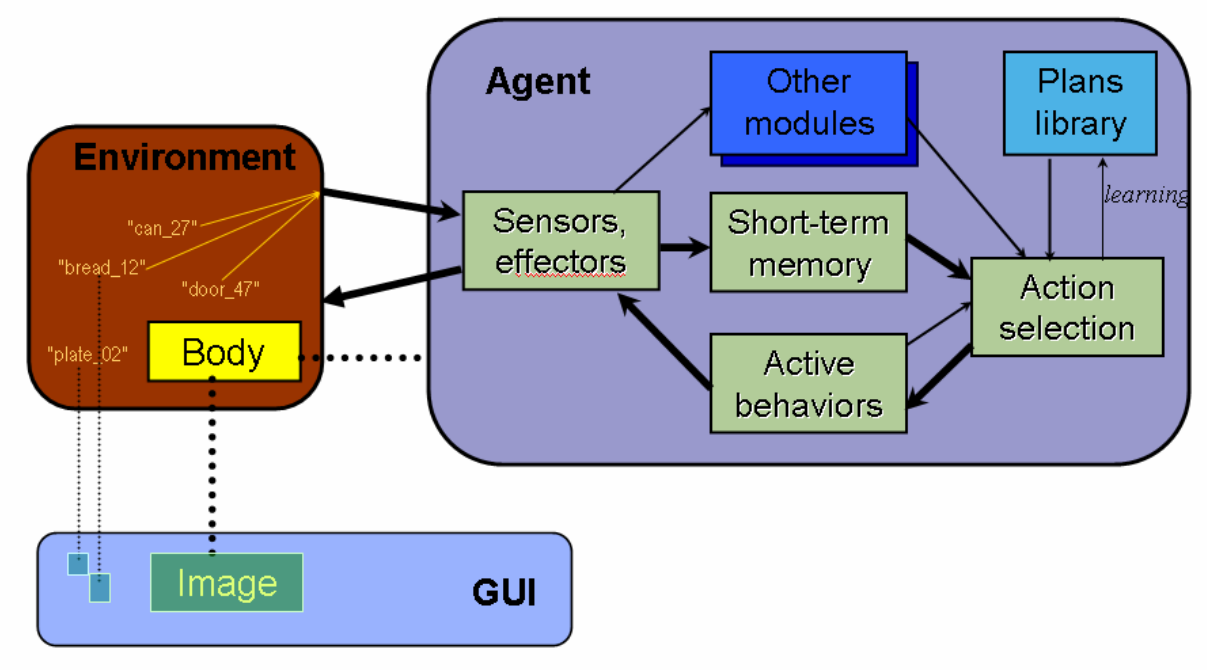

Fig. 2: An overall schema of an agent driven by reactive rules or finite state machines. The main part, which is denoted as "action selection", performs action selection based on facts stored in the short-term memory, reactive plans and the currently active behaviour. The selection can be also influenced by other modules, e.g. drives or emotions.

Connectionists approach

More or less, *connectionist networks* like *artificial neural networks* (e.g. Grand et al., 1997), *free-flow hierarchies* (e.g. Tyrrell, 1993; de Sevin and Thalmann, 2005) or *spreading activation networks* (Maes, 1991) can be also thought of as dynamic planning approaches, although these are more likely to incorporate adaptation as well. But at any instant, they produce a single action as a consequence of their current state and the state of the world they perceive.

Basically, a connectionist network comprises a set of simple units which are interconnected somehow. Every unit has several input links that feed the unit with an abstract 'activity' and output links that propagate the activity to following units. Each unit itself works as the activity transducer. Typically, there is an input set of units, which receives input from perceptual stimulation, memory or drives, and an output set of units, which select an action to be performed.

The primary advantages of connectionist networks are that they are generally adaptive, and can often learn to fine tune their behaviour, although in some circumstances this can be a problem rather than a solution. Since action selection is more naturally continuous in response to continuous input, resulting behaviour can be smoother than behaviour produced by naively implemented discrete if-then rules and FSMs, although damping in such systems is also possible through the use of separate mechanisms. Finally, some developers prefer being able to specify all possible connections between senses and actions, so that they can easily indicate through weights whether there senses are prescriptive or proscriptive indicators for a behaviour (Tyrrell, 1993; Maes, 1991). These methods also have several disadvantages. Describing sequential behaviour with no external context shift through a network is very difficult, requiring extra recurrent connections and special nodes dedicated to historic state.

Networks are best used for problems that can be solved purely reactively, which are unfortunately rare in action selection. Also, the more complicated a network is, the lower the probability that machine learning can successfully find optimal weights for it, because of the combinatorial complexity of exploring the interactions of too many unrelated variables. This is ironic since adaptation should in theory save development time, but it is empirically true (Tyrrell, 1993).

Connectionists techniques can be also combined with FSMs or if-then rules – see Champandard (2003) for example. Alternatively, connectionist elements can be incorporated within modules that provide the primitive acts the action selection arbitrates between – see Bryson (2001) for examples of this.

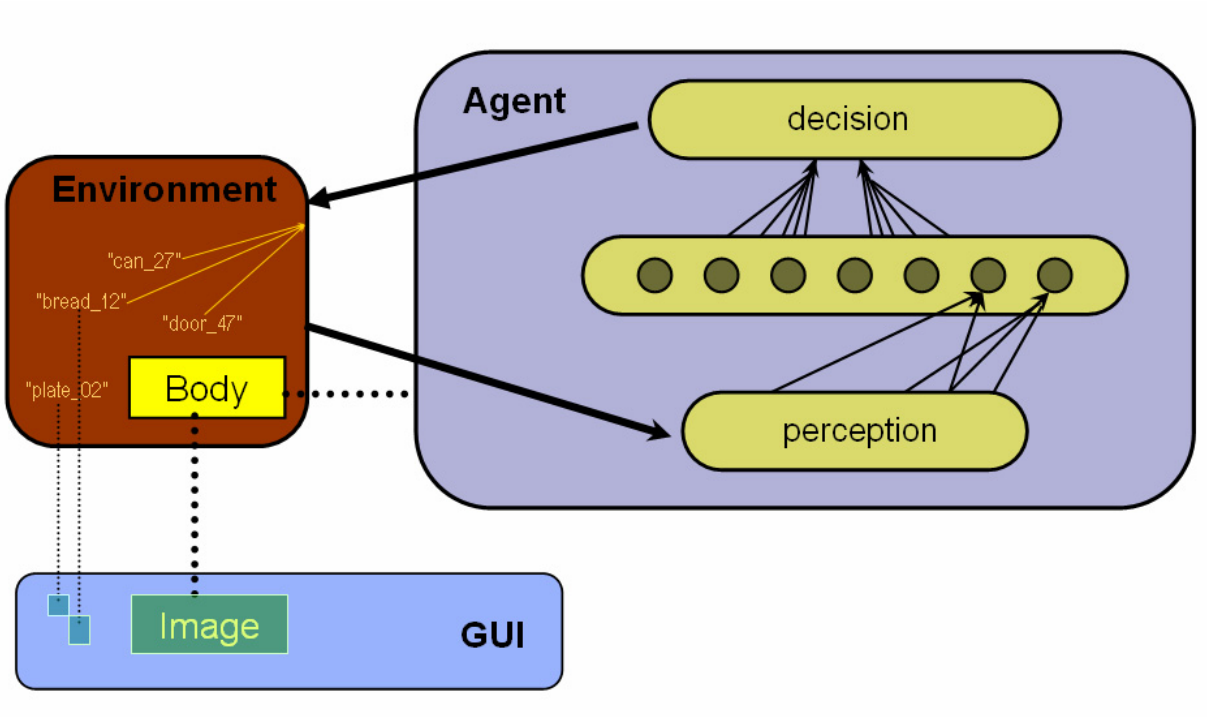

Fig. 3: An overall schema of a connectionist agent. Input units, which are stimulated externally, generate activity, which is propagated throughout the network. Output units select the action. The network itself works also as a memory.

Navigation and Steering

Navigation is one of the most fundamental problems embodied agents face. *Steering rules* are a special reactive technique often used for some of the navigation problems, primarily that concerning flocks or herds of agents. It is based on superposition of attractive and repulsive forces that effect on the agent. Steering is based on the original work of Reynolds (1987).

By means of steering, one can achieve a simple form of:

- navigating towards a goal navigation
- obstacle avoidance behaviour
- wall or path following behaviour

- fleeing enemies and avoiding predators
- and coordinated behaviour (non-interference) by crowds.

The advantage of steering is that it is computationally very efficient. In computer games, hundreds of soldiers can be driven by this technique. In cases of more complicated terrain (e.g. a closed-space in a building), however, steering must be combined with path-finding, which is a form of planning.

Hybrid architectures

Finally, as we have already mentioned, many researchers combine elements of various approaches. In general, this combination will be hierarchical – for example, a neural network may perform the ultimate action selection but its nodes might govern short dynamic-plan scripts, or a symbolic system may reason about top-level goals but rely on a reactive system to execute the associated behaviour, or an FSM might refer to states that govern their system through adaptive controllers containing fuzzy logic.

Goal driven architectures

In these architectures, an agent's behaviour is typically described by a set of *goals*. Each goal can be achieved by a process or an activity. This in turn can be described, for example, by a prescripted *dynamic plan* (see above). The agent then just decides which process to carry out in order to accomplish a given goal. The plan can be expanded to include subgoals, which makes the process recursive and hierarchical. Generally, such plans use condition-action rules. These architectures can be hybrid, since in deciding which process to carry on to accomplish a goal a planning technique can be used.

Classical examples of goal driven architectures are implementable refinements of Michael Bratman's framework of practical reasoning (Bratman, 1987), better known now as the Belief-Desire-Intention (BDI) architecture (e.g. Brom et. al, 2006; Huber, 1999; JACK, 2006).

Layer architectures

Towards the end of the 1990s, several publications proclaimed that *three-layered hybrid architectures* had been proven the best strategy for controlling autonomous robots. The three layers were a symbolic reasoning systems at the top, dynamic planning systems in the middle, and a lowest layer consisting of behaviour modules – small units for translating perception directly to action (see Bryson, 2000 for a review). On the one hand, as of 2006, the main three-layered systems (PRS and 3T) do not appear to have been supported or maintained for at least five years. Modifications to ACT-R and Soar made so that these architectures could operate robots are somewhat similar to the three-layered approach though (Laird and Rosenbloom 1996). Bespoke layered solutions also continue to appear. For example, in computer game F.E.A.R (Orkin, 2006), there is used a combination of a simplified planning with a dynamic planning system and classical pathfinding and steering techniques.

See Wooldridge (1999) for general overview of layered architectures.

Anytime planning

Anytime planning is a useful extension of classical planning that interleaves computation of a plan with its execution (Dean and Boddy, 1988). Of course, anytime planning still suffers from combinatorial complexity, but the algorithms are carefully designed such that a best guess at a solution is always available, but if more time is given a solution that is at least as good, probably better, will be offered. Gat (1998) points out how this can be incorporated into a hybrid action-selection architecture: while the plans for the system can be continuously improved, the current best plan is always available to be applied dynamically.

An example of a relatively recent anytime planning architecture is Excalibur (Nareyek, 2005), a research project led by Alexander Nareyek. The architecture is based on structural constraint satisfaction, which is an advanced artificial intelligence technique. Recently, however, the project has been discontinued.

Conclusions

Action selection is a fundamental part of intelligent behaviour, yet no consensus has so far emerged for the best mechanism to be used for AI systems, or even for particular classes of problems. Nevertheless, there is a large existing body of work which can be extended, and there does seem to be convergence on the technique of dynamic planning for real-time commercial applications with little available CPU. On the other hand, many animat researchers favour evolved weights for neural networks in the hope that this approach will eventually scale to be a useful semi-automated programming system. In this overview we have concentrated on practical techniques for action selection, but there is also a good deal of exciting work being done in neurological modelling of action selection in animal brains.

References

[Bratman, 1987] Bratman, N.: *Intention, plans, and practical reason*. Cambridge, Mass: Harvard University Press (1987)

[Brom et al., 2006] Brom, C., Lukavský, J., Šerý, O., Poch, T., Šafrata, P.: Affordances and level-of-detail AI for virtual humans. In: *Proceedings of Game Set and Match 2*, Delft – and also: http://urtax.ms.mff.cuni.cz/ive [8th of May, 2006]

[Bryson, 2003] Bryson, J. J.: Action Selection and Individuation in Agent Based Modelling. In: D. L. Sallach and C. Macal (eds.): *Proceedings of Agent 2003: Challenges in Social Simulation*}, Argonne National Laboratory, Argonne, IL, USA. (2003) 317 – 330

[Bryson, 2001] Bryson, J. J.: *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. PhD thesis, Massachusetts Institute of Technology (2001)

[Bryson, 2000] Bryson, J. J.: Cross-Paradigm Analysis of Autonomous Agent Architecture. In: *Journal of Experimental and Theoretical Artificial Intelligence*, **12**(2) (2000) 165-190

[Bryson and Stein, 2000] Bryson, J. J., Stein, L. A.: Architectures and Idioms: Making Progress in Agent Design. In: C. Castelfranchi and Y. Lespéerance (eds.): *The Seventh International Workshop on Agent Theories, Architectures, and Languages*, Springer (2000)

[Champandard, 2003] Champandard, A. J.: *AI Game Development: Synthetic Creatures with learning and Reactive Behaviors*. New Riders, USA (2003)
– and also: http://aigamedev.com/ [8th of May, 2006]

[Dean and Boddy, 1988] Dean, T., Boddy, M. An Analysis of Time-Dependent Planning, In: *Proceedings of the Seventh National Conference on Artificial Intelligence* (AAAI-88), AAAI Press/MIT Press, Saint Paul, Minnesota, USA (1988) 49 – 54

[Nareyek, 2005] Nareyek, A. Excalibur project. Project homepage: http://www.ai-center.com/projects/excalibur/index.html [8th of May, 2006]

[Gallistel et al., 1991] Gallistel, C. R., Brown, A. L., Carey, S., Gelman, R., Keil, F.C.: Lessons From Animal Learning for the Study of Cognitive Development. In: Carey, S., Gelman, R. (eds.): *The Epigenesis of Mind*. Lawrence Erlbaum, Hillsdale, NJ (1991)

[Gat, 1998] Gat, E. Three-Layer Architectures. In: Kortenkamp, D., Bonasso, R. P., and Murphy, R.: *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA (1998) 195 – 210

[Grand et. al, 1997] Grand, S., Cliff, D., Malhotra, A.: Creatures: Artificial life autonomous software-agents for home entertainment. In: Johnson, W. L. (eds.): *Proceedings of the First International Conference on Autonomous Agents.* ACM press (1997) 22 – 29
– and also: http://creatures.wikia.com/wiki/Creatures_Wiki_Homepage/ [8th of May, 2006]

[Huber, 1999] Huber, M. J.: JAM: A BDI-theoretic mobile agent architecture. In: Proceedings of the Third International Conference on Autonomous Agents (Agents'99). Seatle (1999) 236 – 243
– and also: http://www.marcush.net/IRS/irs_downloads.html [8th of May, 2006]

[Isla, 2005] Isla, D.: Handling complexity in Halo 2. In: *Gamastura online*, 03/11 (2005)
– and also: http://www.gamasutra.com/gdc2005/features/20050311/isla_pfv.htm [8th of May, 2006]

[JACK, 2006] JACK toolkit. Agent Oriented Software Group. Project homepage: http://www.agent-software.com/shared/home/ [4th of August, 2006]

[Laird & Rosenbloom 1996] Laird, J.E. and Rosenbloom, PS: The Evolution of the Soar Cognitive Architecture. In D. M. Steier and T. M. Mitchell (eds.), *Mind Matters*, Erlbaum (1996)

[Maes, 1991] Maes, P.: The agent network architecture (ANA). In: SIGART Bulletin, 2 (4) (1991) 115–120

[Orkin, 2006] Orkin, J.: 3 States & a Plan: The AI of F.E.A.R. In: *Game Developer's Conference Proceedings* (2006)
– and also: http://web.media.mit.edu/~jorkin/ [14th of July, 2006]

[Prescott, 2007] Prescott, T. J.: Forced moves or good tricks? Landmarks in the evolution of neural mechanisms for action selection. In: *Adaptive Behavior: Special issue on Mechanisms of Action Selection* **15**(1) in press.

[Reynolds, 1987] Reynolds, C. W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. In: *Computer Graphics*, **21**(4) (SIGGRAPH '87 Conference Proceedings) (1987) 25-34.

[de Sevin and Thalmann, 2005] de Sevin, E. Thalmann, D.: A motivational Model of Action Selection for Virtual Humans. In: *Computer Graphics International* (CGI), IEEE Computer SocietyPress, New York (2005)

[Soar project, 2006] Soar project. University of Michigan. Project homepage: http://sitemaker.umich.edu/soar [8th of May, 2006]

[Softimage/Behavior, 2006] Softimage/Behavior product. Avid Technology Inc. Product homepage: http://www.softimage.com/products/xsi/pricing_and_packaging/advanced/behavior/ [14th of July, 2006]

[Tyrrell, 1993] Tyrrell, T.: Computational Mechanisms for Action Selection. Ph.D. Dissertation. Centre for Cognitive Science, University of Edinburgh (1993)

[van Waveren, 2001] van Waveren, J. M. P.: *The Quake III Arena Bot*. Master thesis. Faculty ITS, University of Technology Delft (2001)

[Wooldridge, 2002] Wooldridge, M.: *An Introduction to MultiAgent Systems.* John Wiley & Sons (2002)

Sources for further reading

[Brom, 2006] Brom, C.: Slides on a course on action selection of artificial beings, on-line. Charles University in Prague: http://ksvi.mff.cuni.cz/~brom/teaching.html#umelebytosti [8th of May, 2006]

[ASM, 2006] The University of Memphis: Agents by action selection, on-line. http://www.msci.memphis.edu/~classweb/comp7990/fall2002/action.htm [8th of May, 2006]

[Prescott et al., 2006] Prescott, T. J, Bryson, J. J. and Seth, A.: Modelling Natural Action Selection, a special issue of *Philosophical Transactions of The Royal Society, B: Biology.* In press.

[Wooldridge, 1999] Wooldridge, M. Intelligent Agents. In: Weiss, G. (eds.): *Multiagent Systems.* The MIT Press (1999)
– and also: http://www.csc.liv.ac.uk/~mjw/pubs/mas99.pdf [14th of July, 2006]