Investigating the Complex with Virtual Soccer

Itsuki Noda and Ian Frank

Complex Games Lab
Electrotechnical Laboratory
Umezono 1-1-4, Tsukuba
Ibaraki, Japan 305
Email: {noda,ianf}@etl.go.jp

Abstract. We describe Soccer Server, a network-based simulator of soccer that provides a virtual world enabling researchers to investigate the complex system of soccer play. We identify why soccer is such a suitable domain for the creation of this kind of virtual world, and assess how well Soccer Server performs its task. Soccer Server was used in August 1997 to stage the simulation league of the first Robotic Soccer World Cup (RoboCup), held in Nagoya, Japan. This contest attracted 29 software soccer teams, designed by researchers from ten different countries. In 1998, an updated version of Soccer Server will be used to stage the second RoboCup in France, coinciding with the real World Cup of football.

1 Introduction

The rules of soccer, or association football, were formulated in 1863 in the UK by the Football Association, and the game has since become one of the most widely played in the world. The popularity of the game is illustrated by the following passage:

> Contrary to popular belief, the world's greatest sporting event in terms of prolonged, worldwide interest is not the Olympic Games. Rather, it is the World Cup of football, which, like the Olympics, is held just once every four years and is played out over a period of two weeks or more. The United States hosted this global spectacle in 1994—and what a spectacle it turned out to be. The championship game between Brazil and Italy was witnessed live by a crowd of over 100,000 people in the Rose Bowl in Pasadena, California, and by a crowd of at least a billion on television the world over.

We quote this description in our introduction not simply because it illustrates the popularity of soccer, or because the publication of this paper coincides with France-98, the successor to the American World Cup. Rather, we are just as interested in the source of the excerpt — not a sports book or newspaper, but the opening paragraph of the opening chapter of a book on science: *Would-be Worlds*, by John Casti [Casti 97a].

Casti's purpose in describing the football World Cup is to introduce the notion of *complex systems*. Football is an excellent example in this context because it is not possible to understand the game by analysing just individual players. Rather, it is the interactions in the game, both between the players themselves and between the players and the environment, that determine the outcome. This is the essence of complex systems: they resist analysis by decomposition.

We share Casti's belief that the arrival of cheap, powerful widespread computing capability over the past decade or so has provided us with a tool for studying complex systems as complete entities. Casti himself suggests that such computer simulations (virtual worlds, would-be-worlds) "play the role of laboratories for complex systems", which, in contrast to more conventional science laboratories "allow us to explore information instead of matter" [Casti 97b]. It is to provide this kind of laboratory for research on complex systems that we have developed a simulation of the game of soccer. This system, Soccer Server, enables a soccer match to be played between two teams of player-programs, and has already been used to stage a 29-team contest at the First Robotic Soccer World Cup (RoboCup), in Nagoya, Japan. In this paper, we describe Soccer Server, identifying in particular why soccer is such a suitable choice as a simulation domain. We also take care to evaluate the success of Soccer Server, demonstrating that the properties of *fidelity*, *simplicity*, *clarity*, *bias-free*, and *tractability* are effective measures for assessing this kind of simulation.

2 Soccer as a Domain for Modelling

There were many reasons behind the choice of soccer for our simulator. Initially, of course, there was the appeal that soccer is fun and easily understood by large numbers of people. More important than this, however, was the significant challenge offered by trying to formalise and understand the domain.

2.1 Soccer as a Complex System

To illustrate the difficulty of the soccer domain, let us return to the discussion of complex systems that we started in the Introduction, and to Casti's book, *Would-be Worlds*. In discussing how a theory of the complex might be formalised, Casti introduces a number of characteristics he describes as the "key components" of complex, adaptive systems. Briefly, these are (1) a medium-sized number of agents, that (2) are intelligent and adaptive, and (3) only have local information.

It should not be too hard to convince the reader that the game of soccer is a very good fit for these properties. Soccer has 22 agents. This falls comfortably between Casti's examples of systems with large numbers of agents (galaxies, which have enough agents to be treated statistically), and small numbers of agents (conflicts between two superpowers). The agents in soccer are also intelligent and adaptive, with the players on each team striving to perform well together, and to out-manœuvre the opponents. Finally, the information in soccer is clearly limited, as the players can only see in the direction they are facing,

Fig. 1. Window image of Soccer Server

and verbal communication is hampered both by distance and by the importance of concealing intentions from the opponents.

Thus, understanding soccer is a significant challenge: it has the characteristics of a complex system, for which, as Casti emphasises, "there exists no decent mathematical theory" [Casti 97b]. However, as well as being a good example of a complex system, another important factor also influenced our choice of soccer: the feasibility of constructing a simulation. We examine this question below.

2.2 The Nature of Play

One of the foremost tasks in the design of a simulation is the creation of a *model* of the domain in question; it is the implementation of the rules and principles of the model that produces the simulation. In this respect, we found that soccer was a very amenable domain, simply by virtue of being a game.

An excellent study of the nature of play is given by Huizinga [Huizinga 50]. Although primarily interested in the significance of play as a cultural phenomenon, some of the main characteristics of play identified by Huizinga are relevant to our theme here. For instance, the first properties of play identified by Huizinga are that it is voluntary (or, as Huizinga says, "Play to order is no longer play"), and that play is not "ordinary" or "real" life. These properties already suggest some kind of modelling process. Huizinga then notes that "all play is limited in locality and duration" — another feature that makes play a promising candidate for convenient simulation.

Further general properties identified by Huizinga are that "all play has binding rules" and that "play can be repeated and transmitted". The presence of rules means that part of the work of modelling is already done. Indeed, in implementing Soccer Server, we took the simple approach of basing our model primarily on just the rules of the game. The remainder of the Soccer Server model was then formulated to allow maximum scope for the investigation of how the rules of soccer can actually be followed in the best way; that is, how to create a team that can play well. Huizinga's final property of repeatability and transmittability then indicates how our model can be used as an ideal testbed for allowing researchers to investigate and discuss the properties of the domain.

2.3 The Type of Model Represented by Soccer Server

As we suggested above, Soccer Server models the rules of the game of soccer in a way that allows the system to be used as a testbed for research on the nature of the domain. The intention is that different soccer player-control algorithms can be tested against each other to discover which are stronger. Thus, Soccer Server can be viewed as a predictive model of the relative strengths of these algorithms. We agree with [Casti 97a, Page 25] that "the first and foremost test that a model must pass is that it must provide convincing answers to the questions we put to it". Note, though, that the questions we want to answer with Soccer Server are not just which algorithms perform better, but also *why* the better algorithms are superior. This is a consequence of viewing the system as a testbed for research; the ultimate goal is to advance the theory of complex systems.

We should also note that although this paper concentrates on the virtual world created by Soccer Server, the RoboCup tournaments themselves also include competitions for teams of real robots. In our view, this link is very important, since there are many results that show that real-world experiments can produce results that could not be expected from simulation alone (*e.g.*, see [Thompson 97], [Brooks 86]). With this caveat said, then, let us move on to take a detailed look at our implementation of a model of soccer.

3 Soccer Server Itself

Soccer Server enables a soccer match to be played between two teams of player-programs (possibly implemented in different programming systems). A match using Soccer Server is controlled using a form of client-server communication. Soccer Server provides a virtual soccer field (such as the one we presented in Figure 1) and simulates the movements of players and a ball. A client program

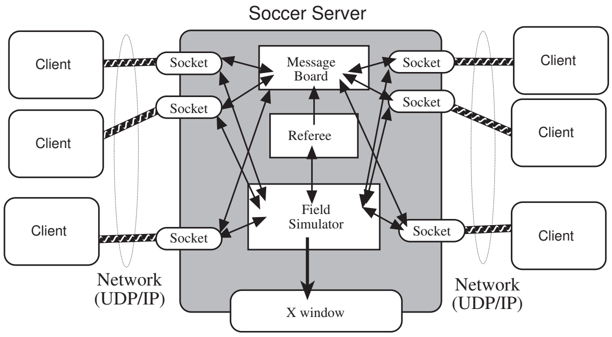

Fig. 2. Overview of Soccer Server

can provide the 'brain' of a player by connecting to the Soccer Server via a computer network (using a UDP/IP socket) and specifying actions for that player to carry out. In return, the client receives information from the player's sensors.

Figure 2 gives an overview of how the Soccer Server communicates with clients. The three main modules in the Soccer Server itself are:

1. **A field simulator module**. This creates the basic virtual world of the soccer field, and calculates the movements of objects, checking for collisions.
2. **A referee module**. This ensures that the rules of the game are followed.
3. **A message-board module**. This manages the communication between the client programs.

Below, we give an overview of each of these modules. A more detailed description of the Soccer Server can be found in [Noda *et al* 98]. Also, a large collection of related material, including sources and full manuals, is maintained at the Soccer Server home page: `http://ci.etl.go.jp/~noda/soccer/server`.

3.1 Simulator Module: The Basic Virtual World

The soccer field and all objects on it are 2-dimensional. This means that, unlike in real football, the ball cannot be kicked in the air and the players cannot make use of skills such as heading or volleying. The size of the field, measured in the notional internal equivalent of metres, is $105\text{m} \times 68\text{m}$. On the field, there are objects that move (the players and the ball) and objects that are stationary (flags, lines and goals). The stationary objects serve as landmarks that clients can make use of to determine their player's position, orientation or speed and direction of movement. Clients are not given information on the absolute positions of their own players, but only on the relative location of other objects. Thus, the fixed objects are important reference points for the client programs' calculations.

All communication between the server and each client is in the form of ASCII strings. Therefore, clients can be realized in any programming environment on any architecture that has the facilities of UDP/IP sockets and string manipulation. The protocol of the communication consists of:

- **Control commands**: Messages sent from a client to the server to control the client's player. The basic commands are `turn`, `dash` and `kick`. Communication is conducted through the `say` command, and a privileged goalie client can also attempt to `catch` the ball. The `sense_body` command provides feedback on client status such as stamina and speed, and `change_view` selects a trade-off between quality of visual data and frequency of update.
- **Sensor information**: Messages sent from the server to a client describing the current state of the game from the viewpoint of the client's player. There are two types of information, visual (`see`) and auditory (`hear`) information.

Soccer Server is a discrete simulation of continuous time. Thus, both the control commands and the sensor information are processed within a framework of 'simulator steps'. The length of the cycle between the processing steps for the control commands is 100ms, whereas the length of the step cycle for the sensor information is determined by the most recent `change_view` command issued by a client (we discuss the timing of these steps in more detail in §4). Note that all players have identical abilities (strength and accuracy of kicking, stamina, sensing) so that the entire difference in performance of teams derives from the effective use of the control commands and sensor information, and especially from the ability to produce collaborative behaviour between multiple clients.

As a final feature, when invoked with the `-coach` option, the server provides an extra socket for a privileged client (called a coach client) that has the ability to direct all aspects of the game. The coach client can move all objects, direct the referee module to make decisions, and announce messages to all clients. This facility is extremely useful for tuning and debugging client programs, which usually involves repeated testing of the behaviors of the clients in many situations. An extension being considered for further versions of Soccer Server is a modified version of the `-coach` option that allows teams to include a twelfth client that has a global view of the game and can conduct sideline coaching during play by shouting strategic or tactical advice to players.

3.2 Referee Module: Playing by the Rules

The referee module monitors the game and, as in real soccer, regulates the flow of play by making a number of decisions based on the rules of the game (announcing goals, throw-ins, corner kicks, and goal kicks, and time-keeping). In the first RoboCup tournament, the referee made no decisions about fouls. The new version of Soccer Server, however, introduces an offside rule, adding to the possible judgments of the referee. Remaining fouls, like 'obstruction', are difficult to judge automatically as they concern players' intentions. The server therefore also includes an interface allowing a human user to instruct the referee to award a free kick. Since such decisions about fair play actually call for significant understanding of the game, future versions of Soccer Server may include a dedicated socket for referee clients, thus encouraging research on automatic refereeing.

3.3 Message Board Module: Communication Protocol

The message board module is responsible for managing the communication between the players. A basic principle of Soccer Server is that each client should control just one player, and that communication between individual clients should only be carried out via the `say` and `hear` protocols introduced in §3.1.

When a `say` command is issued, the message is broadcast to *all* clients immediately as auditory information. In early versions of Soccer Server, teams could use this communication to sidestep the local nature of information in the game by, for example, having each player broadcast his own location and interpretation of the game at each time step. Another possible strategy was to use one client in a team (*e.g.*, a 'captain') to direct and inform all the others. In the current Soccer Server, however, not all the other clients are guaranteed to hear all the broadcast information, since there is a maximum range of communication of 50m, and any player can only hear one message from each team during each two simulation cycles. Also, the length of the message itself is restricted. Thus, effective and efficient use of the `say` command is encouraged, imparting information only when it is useful and timely.

Note that the server can connect with up to 22 clients, but that to facilitate testing it is possible for a single program to control multiple players (by establishing multiple socket connections). In a competitive situation such as RoboCup, such programs are only permitted on the understanding that the control of each player is separated logically. In practice, it is also not technically difficult to cheat the server and establish inter-client communication outside the server, so the 'one-client one-player' rule is effectively a gentlemans' agreement.

3.4 Uncertainty

In order to reflect the nature of the real world, the server introduces various types of uncertainty into the simulation, as follows:

- Noise added to the movement of objects. The amount of this noise increases with the speed of the object.
- Noise added to command parameters. Small random numbers are added to the parameters of commands sent from clients.
- Limited command execution. The server executes only one command for each player in each simulation cycle. Each client program can get feedback on how many commands its player has executed via use of the `sense_body` command (*successful* execution must of course be monitored by the clients themselves; for instance, a `dash` has no effect if a player's path is blocked).
- Inexact sensing. The further an object, the less reliable the information returned about it from the server.

The presence of these uncertainties re-enforces the importance of robust behavior, and of the reactive monitoring of the outcomes of a player's actions.

4 Assessing the Soccer Server

How can we meaningfully assess the Soccer Server? In §2.3 we suggested that the main test of a good model is whether it provides answers to the questions we ask of it. Thus, since we constructed Soccer Server to enable the investigation of soccer, we could simply examine the quality of the conclusions learned about agent behaviour in the domain. However, a more fundamental review is also possible. For instance, [Casti 97a, Page 175-176] summarises several properties that can be used to assess models and simulations:

- **Fidelity**. The model's ability to represent reality to the degree necessary to answer the questions for which the model was constructed;
- **Simplicity**. The level of completeness of the model, in terms of how "small" the model is in things like number of variables, complexity of interconnections among sub-systems, and number of *ad hoc* hypotheses assumed;
- **Clarity**. The ease with which one can understand the model and the predictions/explanations it offers for real-world phenomena.
- **Bias-free**. The degree to which the model is free of prejudices of the modeler having nothing to do with the purported focus or purpose of the model.
- **Tractability**. The level of computing resources needed to obtain the predictions and/or explanations offered by the model.

Note that these qualities are more subtle than simply assessing whether a model faithfully captures all aspects of the phenomenon it represents. Let us examine the results of evaluating Soccer Server against each of the criteria.

4.1 Fidelity

One of the most important considerations during the development of Soccer Server was the level of abstraction for representing the client commands and the sensor information. One possibility was a low-level, physical description, for example allowing power values for drive motors to be specified. However, it was felt that such a representation would concentrate users' attention too much on the actual control of a players' actions, relegating true investigation of the multi-agent nature of team-play to the level of a secondary objective. Further, it is difficult to design a low-level description that is not implicitly based on a specific notion of robot hardware; for example, control of speed by drive motors is biased towards a physical implementation that uses wheels. On the other hand, a more abstract representation, maybe using tactical commands such as `pass-ball-to` and `block-shoot-course`, would produce a game in which the real-world nature of soccer becomes obscured, and in which the development of soccer techniques not yet realised by human players becomes problematic. Thus, our representation — using basic control commands such as `turn`, `dash`, and `kick` — is a compromise. To make good use of the available commands, clients will need to tackle both the problems of control in an incomplete information, dynamic environment and also the best way to combine the efforts of multiple players. Thus, we believe that Soccer Server achieves our goal of providing a simple test-bench with significant real-world properties.

The choice of abstraction level is also relevant to a further question that often occupied us during the design of Soccer Server: whether the simulator should be designed to model human soccer play or robot soccer play. This question has many facets, but in general the solution adopted in the Soccer Server is to be faithful to human soccer whenever possible. The justifications for this include the reasoning that real-world soccer is more immediately understood (and more attractive) to the casual observer, and that robot technology can change, whereas the nature of human soccer players is largely constant. Also, the closer the simulation to real-world soccer the more likely that knowledge acquisition from human expertise (*e.g.*, see [Frank 97]) will be directly applicable.

Of course there are differences between real soccer and the model represented by the server, most notably the 2-dimensional nature of the simulation. Also, the simulation parameters are tuned to make the server as useful as possible for the evaluation of competing client systems, rather than to directly reflect reality (for example, the width of the goals is 14.64m, double the size of ordinary goals, because scoring is more difficult when the ball cannot be kicked in the air). This question of whether human soccer is being directly simulated is also important in the context of the overall RoboCup challenge, which we discuss further below.

4.2 Simplicity

The ultimate goal of RoboCup, as described in [Kitano *et al* 97a], is "to develop a robot soccer team that can beat the Brazil world-cup team". Since this goal is well beyond current technologies, the RoboCup initiative defines a series of well-directed subgoals that are achievable in the short and mid-term. Implementing clients for Soccer Server is one of these challenges. The objective of RoboCup is that, as the level of sophistication of existing technology improves, the simplicity of the simulation should be altered appropriately.

One of the primary concerns about simplicity is the timing of the simulation cycles that govern the processing of commands and sensor information. The sensor information received by clients is rich, but the actions are simple. This is an argument for making the time between information updates longer than the time between action execution (*i.e.*, having an action processing cycle that is shorter than the information sensing cycle). If the information sensing cycle becomes too long, the ability to react to the opponents' actions becomes hampered. On the other hand, if the information sensing cycle becomes too short, the benefits of attempting to learn and predict the opponents' actions are diminished. The current lengths of these time steps are a compromise intended to strike a balance between these two conflicting goals.

4.3 Clarity

A large number of researchers have already used Soccer Server (we have already mentioned that the first RoboCup contest featured 29 teams from ten different countries). This is evidence that the workings of Soccer Server can be easily understood. In considering the 'clarity' of the system, though, we also have to take into account the ease with which one can understand the lessons learned through the course of such research. So, we are interested in questions like, "What results have emerged out of the teams produced by the first RoboCup contests?" To some extent, the answer to this type of question is dependent on the efforts of the Soccer Server users themselves. However, the RoboCup initiative requires all the teams entering any tournament to write a paper describing their approach. These papers show that in the original contests, the programs that performed well were simple hard-coded systems with little learning ability. For example, at the Pre-RoboCup'96 tournament held in Japan in November 1996, the winning team was the Ogalets, which essentially relied on constraining each player to stay within a small, pre-determined area of the pitch and to choose between a small number of hard-coded passing directions when they could reach the ball. In the RoboCup-97 tournament held in August 1997, on the other hand, the winning teams were more sophisticated. The winners, from Humboldt University, used case-based reasoning and agent-oriented programming, the runners-up used reinforcement learning, and the third place team used an explicit model of teamwork, based on joint intentions.

As well as demonstrating the clarity of Soccer Server, it should be pointed out that sometimes the users of the system can show almost the opposite: that the system was not actually fully understood by the designers themselves. This happens when bugs are found in the simulation that can be exploited to the advantage of the client programs. One example of such a bug was an error in the implementation of stamina. Generally, the stamina of a player decreases when the player issues `dash` commands, thus limiting the player's ability to make further movements. However, it was found by some users that by using *negative* parameters in this command, a player's stamina could be increased!

A further flaw was found with the version of Soccer Server used for the RoboCup-97 contest, in which a problem with the handling of simulation cycles sometimes allowed players to kick the ball three times in very quick succession, thus imparting three times the normal 'maximum' speed. However, it is in the spirit of the RoboCup contest that such bugs are reported to the designers, so that all users are working from an even footing. This spirit is also an illustration of the bias-free nature of Soccer Server.

4.4 Bias-free

Although developed solely at the Electrotechnical Laboratory, Soccer Server has benefited substantially from the opinions and suggestions of many users. In particular, there is a mailing list dedicated to discussing RoboCup, on which a consensus is usually reached before modifying the model represented by the simulation (this mailing list averaged over 200 messages per month between Dec 1997 and March 1998). To give a feel for the speed of change of the simulator, here is a partial list of some of the features changed since RoboCup-97: the introduction of the `sense_body` command, the ability to `see` the direction that other players are facing, the introduction of off-sides, the introduction of a goalie, and a re-modelling of the implementation of stamina. All these changes are designed to increase properties such as the similarity of the simulation to the human game, the importance of modelling the opponents' play, or the importance of modelling the teamwork being produced by a client's own team-mates.

Further evidence of the bias-free nature of Soccer Server is the presence of a public library of code (the RoboCup Simulation Code Archive, located at `http://www.isi.edu/soar/galk/RoboCup/Libs/`). Researchers can use this code to help reduce the development time of RoboCup clients. Some code is specific to particular situations, whereas some is general enough to provide infrastructure for complete agents (the code for the winning clients from RoboCup-97 is available). In particular, the archives contain the 'libsclient' library, a collection of low-level functions intended to abstract away many details of tasks such as maintaining sockets, parsing the information from the server, and sending commands to the server.

4.5 Tractability

Soccer Server has to simulate a game of soccer and communicate with each client program in real time. The system can run on a PC or a workstation with only modest memory and processor resources (although typically a user will also require at least one further computer to run the teams of clients). A more problematic limitation on Soccer Server in practice is the capacity of the network connecting the system to the clients. A large load on the network can lead to collisions that prevent client commands reaching Soccer Server or information being returned. Indeed, some teams at RoboCup-97 found it necessary to re-calibrate their software to the particular conditions found at the tournament site. However, it is difficult to predict the exact effects of changes in network conditions, so client programs with some flexibility in their interpretation of the state of play are encouraged. In general, this may not be an adverse property of the system; the need to understand adaptation and to cope with only local information were two of the prime qualities of complex systems identified in §2.1.

5 Conclusions: Towards a Theory of the Complex

We have discussed how the game of soccer is a good example of a complex system, well-suited to the task of modelling. We also described and assessed our implementation of Soccer Server, clarifying the nature of the model it represents.

Let us close by noting that the challenges represented by the domain of soccer have recently led to it being proposed as a new standard problem for AI research [Kitano *et al* 97b]. Of course, the notion of a standard problem has long been a driving force for engineering research. Historically, for example, the acceptance of the 'Turing Test' [Turing 50] focused attention on the mimicking of human behavior as a test of machine intelligence. More recently, chess has received significant attention, generating important advances in the theory of search algorithms and search control, as well as motivating cognitive studies into the ways that humans approach the same problems (*e.g.*, see [Levinson *et al* 91]).

Soccer (in particular contrast to chess) is a dynamic, real-time, multi-agent, system with incomplete information. We see Soccer Server as a virtual world that provides a tool for investigating such complex domains. It does this by providing a predictive model of the strengths of software algorithms for controlling agents in these environments. In the words of [Casti 97b], Soccer Server is a "would-be-world" that "has the capacity of serving as a laboratory within which to test the hypotheses about the phenomena it represents". It is our hope that, for complex systems such as soccer, the efforts of researchers using the laboratory of Soccer Server will rectify the situation where "at present, there seems to be no known mathematical structures within which we can comfortably accommodate a description" [Casti 97a, Page 214].

References

[Brooks 86] R. A. Brooks. A robust layered control system for a mobile robot. *Journal of Robotics and Automation*, RA-2(1), 1986.

[Casti 97a] John L. Casti. *Would-be Worlds: how simulation is changing the frontiers of science*. John Wiley and Sons, Inc, 1997.

[Casti 97b] John L. Casti. Would-be worlds: toward a theory of complex systems. *Artificial Life and Robotics*, 1(1):11–13, 1997.

[Frank 97] I. Frank. Football in recent Times: What we can learn from the newspapers. In *First Intl. Workshop on Robocup in conjunction with RoboCup-97 at IJCAI-97*, pages 75–82, Nagoya, Japan, 1997.

[Huizinga 50] Johan Huizinga. *Homo ludens: a study of the play element in culture*. Beacon Press, 1950. ISBN 0–8070–4681–7.

[Kitano *et al* 97a] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, pages 73–85, Spring 1997.

[Kitano *et al* 97b] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of IJCAI-97*, pages 24–29, Nagoya, Japan, 1997.

[Levinson *et al* 91] R. Levinson, F. Hsu, J. Schaeffer, T. Marsland, and D. Wilkins. Panel: The role of chess in artificial intelligence research. In *Proceedings of the 12th IJCAI*, pages 547–552, Sydney, Austr alia, 1991.

[Noda *et al* 98] I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer Server: a tool for research on multi-agent systems. *Applied Artificial Intelligence*. To appear.

[Thompson 97] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In T. Higuchi *et al*, editors, *Proceedings of ICES '96 — The First International Conference on Evolvable Systems: from biology to hardware*, pages 385–400. Springer-Verlag, Berlin, 1997.

[Turing 50] A.M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

This article was processed using the LaTeX macro package with LLNCS style