Automated Performance Assessment and Adaptive Training for Training Simulators with SimSCORM

Leo de Penning MSc, Bart Kappé PhD, Eddy Boot PhD
TNO Defense, Safety and Security
Soesterberg, The Netherlands
leo.depenning@tno.nl, bart.kappe@tno.nl, eddy.boot@tno.nl

ABSTRACT

Performance assessment in training simulators for learning complex tasks is a complex task in itself. It requires monitoring and interpreting the student's behavior in the simulator using knowledge of the training task, the environment, and a lot of experience. Assessment in simulators is therefore generally done by human observers. To capture this process in an automated system is challenging and requires innovative solutions. This paper proposes a new module in TNO's Virtual Instruction platform for automated assessment in simulators that is based on Neural-Symbolic Learning and Reasoning. The module is capable of using existing rules and learning new rules for performance assessment. This is done by observing experts and students performing the training tasks. These rules can be used to automatically assess student performance in a training simulator, to validate existing rules and to support the assessment process. The rules can also be used for adaptive training by applying them backwards (relating student competences to adaptations in simulation and instructions). For training organizations, this provides a quicker, cost-saving and more objective evaluation and efficient training of the student in simulation based training, taking into account both explicit and implicit rules. The module will be developed in a three year research project on assessment in driving simulators for testing and examination and tested in various other domains, like jetfighter pilot training and strategic command and control training.

ABOUT THE AUTHORS

Leo de Penning is a senior software engineer and researcher at TNO Human Factors, and is currently working on new and innovative learning technologies for the Training and Instruction department. He holds a MSc in Computer Science and specializes in Artificial Intelligence and Virtual Reality. Previously he worked as system integrator at Atos Origin and as researcher on Artificial Brain building at StarLab.

Bart Kappé is a researcher at TNO Human Factors, and is involved in R&D projects on specification and validation of training simulators, on simulator displays and on motion cueing in simulators. His current research interests include testing in driving simulators, and the integration of simulators and e-learning (and e-testing) systems.

Eddy Boot is a researcher at TNO Human Factors, and is involved in R&D projects concerning the application of Information and Communication Technology (ICT) to improve learning processes. He holds a PhD in instructional technology and specializes in complex learning and competency-based learning supported by advanced learning technologies.

Automated Performance Assessment and Adaptive Training for Training Simulators with SimSCORM

Leo de Penning MSc, Bart Kappé PhD, Eddy Boot PhD

TNO Defense, Safety and Security

Soesterberg, The Netherlands

leo.depenning@tno.nl, bart.kappe@tno.nl, eddy.boot@tno.nl

INTRODUCTION

Performance assessment of learning complex tasks in training simulators has always been a complex task in it itself. For this reason, this assessment is generally performed by human observers. Performance assessment by automated systems is often limited to simple training tasks, because assessing complex tasks requires the modeling of all interrelations between the information present in the simulation, the training tasks, and the constructs being assessed (e.g., competences). Also, when it comes to more subjective assessments (e.g., how 'safe' is the student performing its task), conventional modeling techniques fall short, as the applied assessment rules are often implicit and difficult to elicit from the domain experts.

We propose a new module for automated assessment as part of the Virtual Instruction platform SimSCORM (Penning *et al.*, 2008). This assessment module will be able to learn new rules from the task description, (real-time) simulation data, related assessment data of domain experts or students, and rules already existing (also called background knowledge). These rules can be presented in a human-readable ('*symbolic*') form, facilitating the validation of the assessment rules and supporting the assessment process.

GLOBAL ARCHITECTURE

To assess complex tasks, the automated assessment module requires models of the training task, the student and human assessors (e.g. teachers, examiners or students) and the interrelations between these models. Therefore it requires real-time interaction with the simulator(s), the student and human assessors, and a description of the training task, a student profile and the simulated environment. SimSCORM provides a generic platform for definition and presentation of simulation based training content and interaction between the content, its users and the simulators based on well-known international standards, like SCORM (ADL, 2004), HLA (IEEE, 2000) and XML (W3C, 1998). Via this platform, the automated assessment module can easily access the objects and attributes in the simulation and acquire information on the student profile and progress.

Figure 1 depicts the automated assessment module (named CogAgent) in the context of SimSCORM. SimSCORM provides a player that presents a SCORM-based training task to the students (and possibly one or more human assessors to train CogAgent) via a Learning Management System (LMS). For this purpose, the player uses several agents that operate in a multi-agent configuration. For example, it uses one or more simulation agents that interact with the simulator(s), an assessment agent that does automated performance assessment and learns new assessment rules from observation, and an instructor agent that presents and monitors SCORM-based training tasks and related objectives. Each agent contains a XML-based working memory, which can be configured from a XML-file and interacts with other agents via SOAP (either locally or remote via a webservice).

SimSCORM Editor

To create a XML configuration file for an agent, the SimSCORM platform contains an editor that extends a SCORM editor with the functionality to define simulation specific data. This data describes for example initial conditions, dynamic behavior, training objectives, and related measurements. Also it can contain existing assessment rules that apply to a specific training task or package. These configuration files can then be embedded in a SCORM package and be referred from the training task, represented by a Sharable Content Object (SCO), as part its launch or suspend data.

Simulation Agents

Simulation agents, depicted in Figure 1 as a single instance called SimAgent, act as interface with external simulator(s) via standard communication protocols, like HLA, or serious games via scripting languages, like LUA (1993).

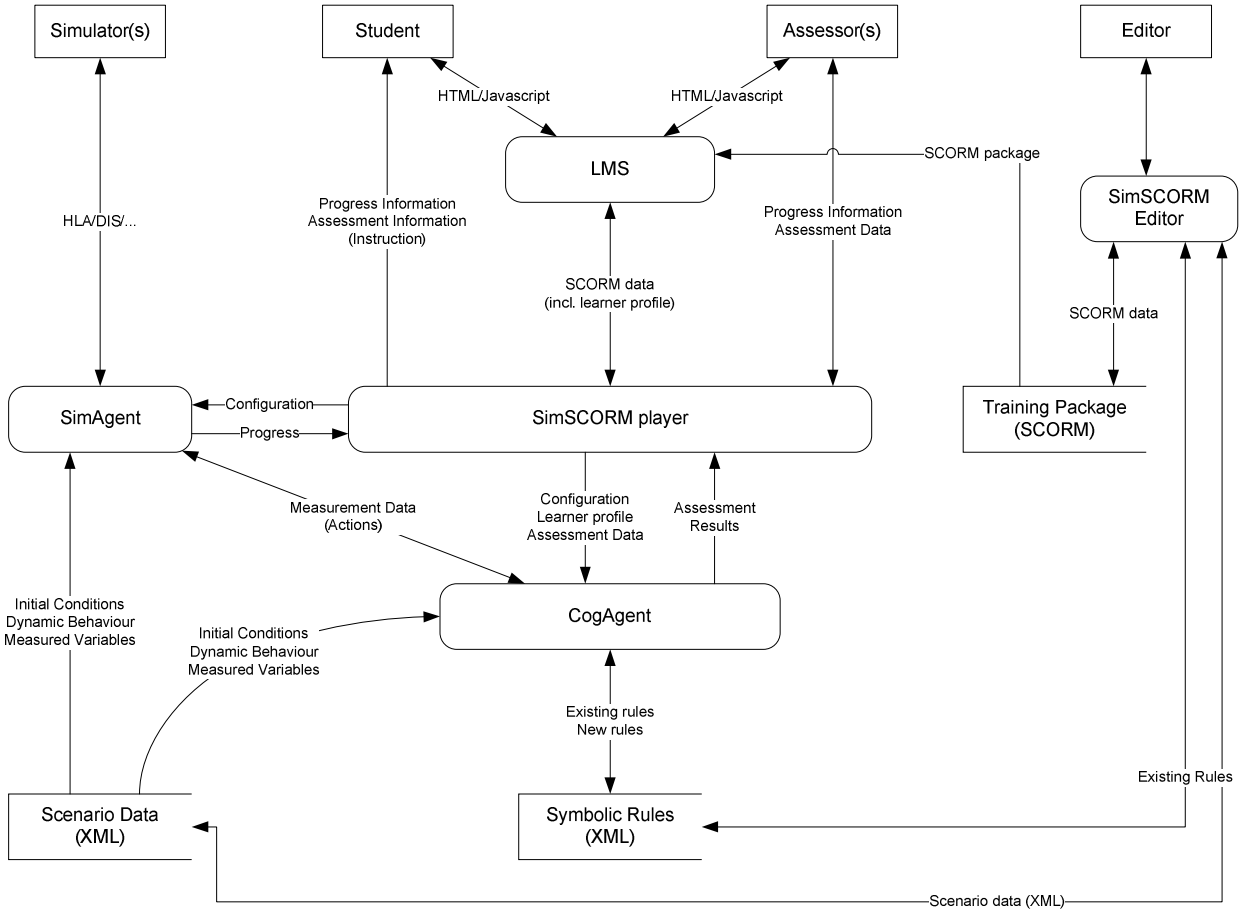

Figure 1. Global architecture of the automated performance assessment module

These agents translate the objects and interactions in the simulation (or serious game) to XML as part of the agent's working memory. For example, SimAgent subscribes to certain objects in the simulation and pre-processes received data to XML elements such that it can be used by CogAgent.

Instructor Agent

The instructor agent, not depicted separately in Figure 1, is the main part of the SimSCORM player and presents and monitors training objectives that are defined for the SCO currently presented to the student. These objectives are defined in the form of SCORM objectives and can be related to data received from the simulation agents, the human assessors, and the LMS. This allows the instructor agent to test student competences, which are related to these objectives, based on the student's actions, a task description, the student's learner profile, previous results, and the learned or predefined rules. Also it can provide feedback and instruction or adapt the behavior of simulation objects based on the outcome of these objectives. This allows the instructor agent to train the student and if desired adapt the training to the student's ability.

Assessment Agent

The assessment agent, depicted in Figure 1 as CogAgent, is configured with information about the training task, measured variables, assessed objectives and existing rules. During execution of the training task, human assessors can provide feedback on the assessed objectives, via the SimSCORM player, which will be presented to CogAgent as short-term evaluations (depicted as assessment data). Based on these evaluations, the student's profile, and information from the simulation agents, CogAgent then determines an overall (or long-term) evaluation for the assessed objectives which will be presented to the students (and assessors) as assessment result. Parallel to this, it uses the simulation data and assessment data to adapt and improve the internal knowledge on assessment rules.

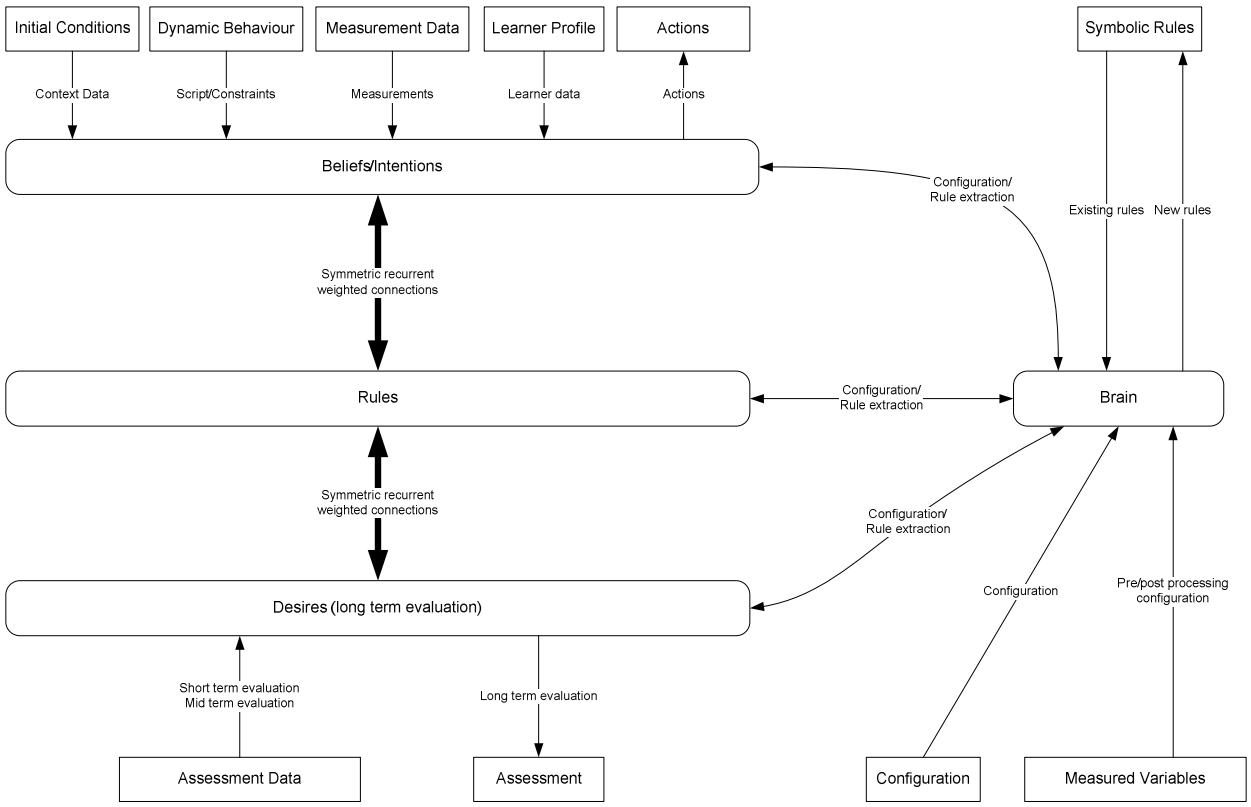

Figure 2. Neural-Symbolic Cognitive Architecture for CogAgent

LEARNING AND REASONING

The assessment agent must be able to learn new rules from observation and existing rules, infer conclusions from these rules and present them in a human readable form. Research on Neural-Symbolic Learning and Reasoning focuses on the integration of learning techniques and architectures from Neural Networks with the symbolic presentation and reasoning techniques in (Fuzzy) Logic Programs (Bader and Hitzler, 2005).

The Neural-Symbolic model proposed for CogAgent is based on the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) (Sutskever *et al.*, 2009) and is depicted in
Figure 2. This partially connected symmetric neural network implements an auto-associative memory of its input layers (called visible layers). CogAgent contains three visible layers that represent its beliefs, desires and intentions (Bratman, 1999). Beliefs are variables related to the training task (initial conditions, dynamic behavior and measured variables) and the student's learner profile. Intentions are variables related to actions (feedback, instructions or adaptive training). And desires are variables related to performance assessment (e.g. training objectives and student competences).

Beliefs and intentions are directly related to the current state of the simulation whereas desires will be related to future states as well using Temporal Difference learning (Sutton, 1988). This technique learns the model to predict a maximum obtainable value for its desires (e.g. overall evaluation scores) based on the current and previous states. Otherwise, the model would only learn to map short-term evaluations, which is not desired in this case.

The hidden layer of the RTRBM is connected to the visible layers with symmetric weighted connections. Each hidden unit in this layer represents a rule or relation between one or more visible units. It also contains recurrent hidden-to-hidden connections that enable the RTRBM to learn the temporal dynamics in the visible layers using an algorithm based on contrastive divergence and back-propagation through time. Using this layer we can infer the posterior probability of beliefs, intentions and desires in relation to the state of other beliefs, intentions and desires and previously applied rules.

Fuzzy Atoms

The assessment rules that CogAgent needs to encode, learn and reason about are relations (or causalities) between XML encoded constructs, which will be called atoms hereafter. An XML based atom describes a belief, intention or desire as a function of measured data from the simulator and/or assessment data from the assessors (or students). In case of training simulators this data is often expressed in both continuous and binary values. Therefore we need to use activation functions in the visible units that can express both. In Chen and Murray (2003), sigmoid functions are introduced that can model binary as well as continuous stochastic functions. These functions contain a 'noise-control' parameter that controls the steepness of the sigmoid function which can be trained. So the actual behavior of a unit is also learned from observation according to the distribution of its input values. We will extend our model with such functions to create a Recurrent Temporal Continuous Restricted Boltzmann Machine (RTCRBM).

Symbolic Rules

To express relations between atoms in symbolic rules we propose to use the temporal propositional logic described in (Lamb *et al.*, 2007). This logic contains several modal operators that extend classical modal logic with a notion of past and future. All these operators can be translated to a form that relates only to the immediate previous time step (denoted by the temporal operator ●). This allows us to encode any rule from this language in the RTCRBM as a combination of visible units (or atoms) and recurrent hidden units that represent applied rules in the previous time step. For example the proposition $\alpha \mathbb{S} \beta$ denotes that a proposition α has been true *since* the occurrence of proposition β. This can be translated to: $\beta \rightarrow \alpha \mathbb{S} \beta$ and $\alpha \wedge \bullet(\alpha \mathbb{S} \beta) \rightarrow \alpha \mathbb{S} \beta$, where α and β are modeled by visible units and $\bullet(\alpha \mathbb{S} \beta)$ is modeled by a recurrent hidden unit.

We extend this logic with the use of equality and inequality formulas to represent the atoms for continuous variables (e.g. $A=x$, $A<x$, etc). Note that the atoms for binary variables can also be represented as $A=true$ or $A=false$, which allows us to handle the outcome of these atoms in the same way as with the continuous atoms. But for readability we will use the classical notion A and $\neg A$.

Due to the stochastic nature of the sigmoid functions used in our model, the atoms can be regarded as fuzzy sets with a Gaussian membership function. This allows us to represent fuzzy concepts, like good and bad or fast and slow or approximations of learned values, which is especially useful when reasoning with implicit and subjective rules. In fact, our model can be regarded as a neural-fuzzy system similar to the fuzzy systems described in (Kosko, 1992) and (Sun, 1994).

Example

Now let's take the training task depicted in the following figure (Figure 3). In this task, the student (depicted by the car with letter T) drives on an urban road and approaches an intersection. In this scenario the student has to apply the yield-to-the-right-rule, which can be regarded as a training objective.

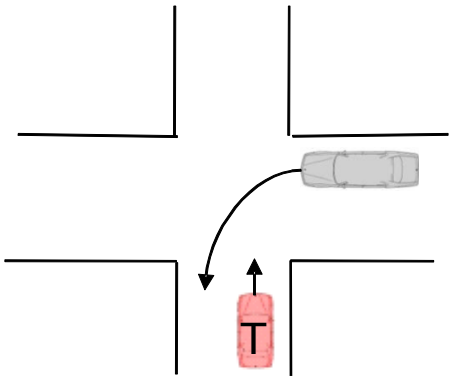

Figure 3. Example training task for driving simulation. The Trainee drives on an urban road, approaching an intersection. The Trainee has to apply the yield-to-the-right-rule.

Using our extended temporal propositional logic, we can describe rules about the conditions, scenario and performance assessment related to this task.

Conditions:

$(Area = urban)$ (1)

$(Weather \geq good)$ (2)

$(Time \geq 6) \wedge (Time \leq 18)$ (3)

Scenario:

$(Speed > 0) \wedge ApproachingIntersection \rightarrow CrossIntersection$ (4)

$ApproachingIntersection \wedge \Diamond(ApproachingTraffic = right)$ (5)

$((Speed > 0) \wedge (HeadingIntersection)) \, \mathbb{S}$ $(DistanceIntersection < x) \rightarrow ApproachingIntersection$ (6)

Assessment:

$ApproachingIntersection \wedge (DistanceIntersection = 0) \wedge (ApproachingTraffic = \text{right}) \wedge \Box(Speed = 0) \rightarrow (Evaluation = good)$ (7)

$ApproachingIntersection \wedge (DistanceIntersection = 0) \wedge (ApproachingTraffic = right) \wedge \Diamond(Speed > 0) \rightarrow (Evaluation = bad)$ (8)

The temporal operator $\mathbb{S}$, used in rule 6, denotes that *ApproachingIntersection* is true when the driver has been driving towards an intersection since a certain distance *x* to an intersection was passed. This rule and the actual value for *x* can be learned from observation by clamping the actual speed, heading and distance to the visible units and the value *true* to the unit for *ApproachingIntersection* when the trainee is approaching the intersection. This can be done by an assessor or the student, but could also be automatically inferred by the model, as explained in the next section.

Rule encoding and extraction

To encode and extract symbolic rules in symmetric connectionist networks, like the RTRBM, Pinkas (1995) describes a generic method that directly maps these rules to the energy function of such networks. Therefore, he describes an extension to propositional logic, called penalty logic that applies a penalty to each rule. This penalty can be regarded as the "certainty" or "reliability" of a rule and is directly related to the weights of the connections between the units that form this rule. To apply the encoding and extraction algorithms of Pinkas (1995) successfully to our model we need extend our temporal propositional logic with the use of penalties. Sun (1994) describes a method to map atoms with classical modal operators to real values (a process called fuzzyfication). We propose to extend this method to create a mapping of atoms and rules with the modal operators used in our model to penalties.

Furthermore, we need to investigate what changes are required to the algorithms to handle the use of equality formulas and continuous variables. For example, we need to prove that it is possible to infer the correct value for unknown continuous variables in a rule via pattern reconstruction based on known values and (previously) applied rules. And to encode and extract rules with inequality formulas (e.g. rules with < or >) we need to be able to transform these to and from rules that contain only equality formulas (rules with = or ≠).

The penalties that are encoded or learned by our model can also be used to rank the rules according to their applicability in a certain context or scenario, giving the students and assessors a ranked overview of the applied rules. Also they allow us to solve ambiguities in the application of rules, by using such a ranking to select the most applicable (or reliable) rule in each case.

ADAPTIVE TRAINING

Because the automated assessment module is capable of learning the relations between student competences, constructs being measured in the simulation and assessments and instructions given during the training, it can be used for adaptive training as well. This means that the results of the assessment module can be used by the instructor agent to adapt the training scenario and/or instructions to make the training task easier or more challenging. This is possible because of the auto-associative nature of the neural-symbolic model that is used in the agent, which allows the learned rules to be applied backwards (i.e. from student competences to changes in behavior of simulation objects and instructions).

RESEARCH AND EXPERIMENTS

The automated assessment module will be developed as part of a three year research project on assessment in driving simulators, carried out by TNO in cooperation with the Dutch licensing authority (CBR), Research Center for Examination and Certification (RCEC), Rozendom Technologies, and ANWB driving schools. A prototype will be ready at the end of 2009 which will be used to do experiments on the ANWB driving simulators used in their drivers training curriculum (see Figure 4). This allows the module to be validated in several scenarios on a large student population using multiple commercial driving simulators. If successful, the module will be used to support the Dutch driver training and examination program.

Figure 4. ANWB Driving Simulators

Figure 5. TNO Mission Simulation Center (MSC)

In parallel, the module will also be tested in other simulation environments, like in TNO's Mission Simulation Center (see Figure 5), for jetfighter pilot training and in Cannibal Hector, for strategic command and control training.

Preliminary results of the experiments will be presented during the paper session.

REFERENCES

ADL (2004). *Sharable Content Object Reference Model (SCORM)*. Advanced Distributed Learning. http://www.adlgov.net.

Bader, S. & Hitzler, P. (2005). Dimensions of neural-symbolic integration - a structured survey. In *We Will Show Them: Essays in Honour of Dov Gabbay, Volume 1. International Federation for Computational Logic*, pages 167-194, College Publications.

Bratman, M.E., (1999). *Intention, Plans, and Practical Reason*. Cambridge University Press.

Chen, H. & Murray, A.F. (2003). Continuous restricted Boltzmann machine with an implementable training algorithm. In *Vision, Image and Signal Processing, IEE Proceedings*, pages 153-158.

IEEE (2000). *IEEE 1516-2000 - Standard for Modeling and Simulation High Level Architecture (HLA)*. Institute of Electrical and Electronics Engineers.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall.

Lamb, L.C., Borges, R.V. & d'Avila Garcez, A.S. (2007). A Connectionist Cognitive Model for Temporal Synchronisation and Learning. In *Proceedings of the Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, pages 827-832.

LUA (1993). *LUA Programming Language*. http://www.lua.org.

Penning, de H.L.H., Boot, E. & Kappé, B. (2008). Integrating Training Simulations and e-Learning Systems: The SimSCORM platform. In *Proceedings of the Conference on Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Orlando, USA.

Pinkas, G. (1995). Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. In *Artificial Intelligence v.77 n.2*, pages 203-247.

Sutskever, I., Hinton, G.E. & Taylor G.W. (2009). The Recurrent Temporal Restricted Boltzmann Machine. In *Advances in Neural Information Processing Systems 21*, MIT Press, Cambridge, MA.

Sun, R. (1994). A neural network model of causality. In *IEEE Transactions on Neural Networks, Vol. 5, No. 4*. pages 604-611.

Sutton, R. (1988). Learning to predict by the methods of temporal differences. In *Machine Learning 3*: pages 9-44, erratum page 377, 1988.

W3C (1998). *Extensible Markup Language (XML) 1.0*. World Wide Web Consortium. http://www.w3.org/tr/xml.