In: P. Hammer, M. Alirezaie, C. Strannegård (eds.), Proc. Artificial General Intelligence 2023, 74-84

Causal reasoning over probabilistic uncertainty

Leonard M. Eberding[1] & Kristinn R. Thórisson[1,2]

[1] Center for Analysis and Design of Intelligent Agents,
Reykjavík U., Menntavegur 1, Reykjavík, Iceland
{leonard20, thorisson}@ru.is
[2] Icelandic Institute for Intelligent Machines

Abstract. A system deployed in the real world will need to handle uncertainty in its observations and interventions. For this, we present an approach to introduce uncertainty of state variables in causal reasoning using a constructivist AI architecture. Open questions of how noisy data can be handled and intervention uncertainty can be represented in a causal reasoning system will be addressed. In addition, we will show how handling uncertainty can improve a system's planning and attention mechanisms. We present the reasoning process of the system, including reasoning over uncertainty, in the form of a feed-forward algorithm, highlighting how noisy data and beliefs of states can be included in the process of causal reasoning.*

Keywords: General Machine Intelligence · Reasoning · Uncertainty

1 Introduction

A major challenge in artificial intelligence (AI) research is the development of systems that can be deployed in the real world and can autonomously adapt to changing circumstances. Additionally, human designers want these systems to be able to generate explanations about why certain interactions were chosen and how the expected state transitions lead to the goal. While deep learning has shown massive advances in the field of data processing and identification of correlated data points, it still lacked to produce a system that is able to adapt to novel circumstances and generate satisfactory explanations [2,1,9]. Reasoning systems, on the other hand, show promising results in both adaptation and explanation generation but often lack the ability to reason over noisy and erroneous data, making deployment in the real world practically impossible, especially when it comes to low-level sensor and actuator precision.

This handling of uncertainty is at the center of research in the field of probabilistic robotics. Under the assumption of sufficiently described mathematical models of the system under control, probabilistic approaches to uncertainty descriptions are used to handle sensor noise and actuator imprecision. The causal

* This work was funded in part by the Icelandic Research Fund (IRF) (grant number 228604-051) and a research grant from Cisco Systems, USA.

reasoning architecture AERA (Autocatalytic Endogenous Reflective Architecture), on the other hand, is designed to extract mathematical models of the environment and the system under control through observation and intervention. By fusing the two approaches, we present a way to overcome limitations in both fields. Learning the transition functions overcomes the necessity to model all dynamics in advance as is done in probabilistic robotics. On the other hand, the limitation to deterministic data streams is lifted from the reasoning system.

Purely probabilistic approaches as used in many Bayesian adaptive methods do not suffice for operation in ever-changing environments, as changes in the distributions of state transitions represented by the probabilistic modeling of the environment can lead to incorrect outcome predictions and erroneous decision making. Instead, we overcome this covariate shift problem [6] by building on hypothesized cause-effect patterns representing invariant world mechanisms.

To address the uncertainty of the in- and output streams, a knowledge representation is needed to enable deductive and abductive reasoning over uncertainty distributions without losing the inspectability of the reasoning processes. Using causal models to describe the state transition itself, we accompany causal models with new probabilistic models that describe the belief propagation under cause-effect structures. Our framework enables reasoning systems to support noisy data streams in both in- and output. Further, we show how such a probabilistic representation can help in describing variables to which the system needs to pay close attention and which do not need to be monitored as closely.

The paper is structured as follows: In the next section, we present related work, focusing on reasoning systems and the transferred principles from probabilistic robotics. In section 3, we present the applied methodology, including a more in-depth analysis of the model structures used in AERA, assumptions that were taken for this work, and the novel approach of including probabilistic models in the reasoning process. Section 4 provides a (simplified) algorithm describing how abductive and deductive reasoning is done. Lastly, in section 5, we discuss our approach in context and how the approach can produce a robust implementation of causal reasoning over probabilistic uncertainty.

2 Related Work

The Autocatalytic Endogenous Reflective Architecture (AERA) [5] follows the constructivist approach to AI, meaning that the system's knowledge base is self-constructing through experiences [7,8]. All knowledge in AERA is non-axiomatic and can be disproven at any point during its lifetime. The reasoning is done on causal models, each representing simple, linearized changes in the environment [7,8]. Coming from the cybernetics side, AERA is designed such that direct control of low-level variables is within its capabilities. Being able to extract linear equation systems from its environment by applying pattern matching on observations and interventions provides valuable functionality for AERA to be used in robotics applications.

Other reasoning systems exist that include falsifiability of their knowledge base, leading to probabilistic and non-axiomatic approaches. One example of this is the Non-Axiomatic Reasoning System (NARS) [11,12]. NARS works under the assumption of insufficient knowledge and resources (AIKR) and is able to reason about non-axiomatic truth statements which can be disproven at any time. Thus, the reasoning process includes the uncertainty of truths. Approaches like the Open-NARS-for-Applications (ONA) [4] provide a framework that can be used in some robotics applications. Another approach to represent uncertainty in the reasoning process are Probabilistic Logic Networks (PLNs) [3]. The first development of PLNs was influenced by NARS but has been extended and nowadays differs from the NARS logic considerably. Based on term logic for inference, PLNs can be used to apply logic operators on probabilistic descriptions of truth values and infer rules through de-, ab-, and inductive reasoning [3]. These approaches, however, do not include probabilistic reasoning over the belief of environment states. Instead, they are more similar to fuzzy logic systems in their description of the uncertainty of the truth of statements.

A major problem is posed by the noisiness and uncertainty of sensors and actuators. No clear, deterministic information exists for the system to reason about. Instead, it is necessary to model the uncertainty of data used in the reasoning process. Probabilistic robotics provides a framework for how to model this uncertainty using Bayesian inference [10]. Uncertainty can be described, propagated, and updated by applying Bayesian statistics and filter methods. For this, the designers of the system define the state-transition functions in advance, which will be applied during run-time. These functions are used to predict new states and observations. Additionally, the uncertainty of these predictions can be calculated as well and updated as new observations come in [10].

In probabilistic robotics, all state-transition functions must be implemented by the designer. We, on the other hand, aim to use Bayesian inference while applying learned causal models. Prediction of future changes to the environment thus becomes a chaining of different transition functions learned by the system, including the prediction of uncertainty.

3 Methodology

In the following, we provide a deeper insight into the methodology of causal reasoning applied in the AERA architecture and OpenAERA in particular before introducing the novel methodology of including probabilistic uncertainty into the reasoning process.

3.1 Autocatalytic Endogenous Reflective Architecture - AERA[1]

Each process observed by OpenAERA is modeled by generating multiple models. These models can be classified as *(anti-) requirement models* and causal models,

[1] See `https://openaera.org` – *accessed Apr. 6, 2023.*

which include *command models*, and *reuse models*. Independent of their type, each causal model represents a left-hand-side (LHS) pattern, a right-hand-side (RHS) pattern, a function that maps LHS to RHS, a function that maps RHS on LHS, and a time interval for which the model holds.

Requirement models describe the constraints under which any causal model (either command or reuse model) is applicable. It gives context to the reasoning system by providing a way that connects observations (or predictions) on the LHS with the instantiation of a causal model on the RHS. This way, the size of the search problem of identifying suitable causal models can be reduced. **Anti-requirement models**, on the other hand, represent conditions under which a causal model does *not* hold. They represent strong requirements and describe hard constraints on the task-environment.

Command models are used to model the direct influence on the environment of executed commands. The LHS of command models is always a command available to the system. The RHS is the change of the environment if the command is executed. Two functions are included in the model. One function is used for forward chaining; it is used to calculate the RHS given the control input and variables passed from the requirement model. The other function is used for backward chaining, representing the inverse of the forward chaining function. For example, a move command which changes the position of OpenAERA in the environment consists of 1) the move command and the associated control input on the LHS, 2) the new position after execution on the RHS, 3) a function that calculates the new position given the old position and the control input, and 4) the inverse of the first function, which can be used to calculate the control input given the current position and the goal position.

Reuse models are used to model similar transitions without an interconnection of state variables. As each model is supposed to model only a very small number of variables to make a reflection on said models possible, reuse models are used to model more complex behaviors. Reuse models have their own requirement models such that complex constraints on complex environment state transitions can be modeled by matching LHS patterns rather than creating massive models responsible for a multitude of calculations. Such a reuse model could, for example, be used to model the changes in an object's state that OpenAERA is holding when a move command is executed. Instead of generating a single model representing the full state change of the system and the object moving simultaneously the same distance, two models are used. The move command model, as previously shown, and a reuse model with its own requirement model.

These models are used to create chains of possible state transitions from the goal to the current state (backward) and from the current state to the goal (forward). OpenAERA is thus able to reason about possible paths to reach the goal by using causal models to represent predicted state changes in the environment.

3.2 Background Assumptions

The following are assumptions underlying the present approach.

1. **Linearity**: One of the underlying architectural concepts of OpenAERA is the step-wise linearization of observed transition functions. Therefore, we assume linearity in all causal models and their uncertainty propagation.
2. **Continuity of variables**: All variables under investigation are assumed continuous in the state space. Reasoning over error-prone non-continuous variables is not part of this work. Other approaches to non-axiomatic reasoning exist for this matter, including other pattern matching and function approximations within OpenAERA.
3. **Normal distribution**: For the sake of simplification, we assume a Gaussian distribution of the measurement and actuator uncertainty. This assumption can be overcome by applying other means to predict uncertainty.
4. **Observation of state variables**: All variables are directly measurable.

3.3 Modeling of probabilistic uncertainty

Any artificial general intelligence (AGI) aspiring system must adapt to novel circumstances to reach a goal/ fulfill a drive under the assumption of insufficient knowledge and resources (AIKR) [12]. This means that it needs to *autonomously* adapt its resource consumption by paying attention to import variables of the task-environment that could influence the reaching of the goal. This includes paying attention to variables/ phenomenons that inflict disturbances on the control problem of transforming the current state to the goal state.

By applying well-known principles from probabilistic robotics, a new model type in OpenAERA will allow it to predict errors in state transitions coming from sensor and actuator imprecision. Its existing attention algorithms are then extended to take into account variables prone to diverge from desired values due to imprecise interventions on the environment. By estimating the posteriori belief of each subsequent state that should be reached during task performance, matching posteriori with a-priori beliefs, and calculating possible overlaps, the system can predict plan divergences. Preemptive measures can then be taken by constraining the control input to achieve intermediate states with a low probability of divergence from the original plan. Probabilistic models work as follows: If, in forward chaining, any causal model represents a noiseless linear function

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{C}\mathbf{u}_{k-1} \tag{1}$$

with $\mathbf{x}$ being the n-dimensional state vector consisting of values of the set of variables $V = \{v_1, v_2, ..., v_n\}$, $\mathbf{C}$ the control matrix and $\mathbf{u}_{k-1}$ the control input.

The accompanying probabilistic model represents the propagation of uncertainty if the model is applied:

$$\mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{C}\mathbf{P}_{control}\mathbf{C}^T + \mathbf{Q}_k \tag{2}$$

with the a priori belief $\mathbf{P}_{k-1}$ of the state $\mathbf{x}$, the posteriori belief $\mathbf{P}_k$, the noise distribution of the command $\mathbf{P}_{control}$ and the process noise $\mathbf{Q}_{k-1}$.

These models are attached to all causal models. The current belief $\mathbf{P}_{k-1}$ is dependent on the source of the input. At the beginning of the reasoning chain, $\mathbf{P}_{k-1}$ represents the noise of the sensor whose observation led to the instantiation of the model. If the input data comes from a prediction, on the other hand, $\mathbf{P}_{k-1}$ represents the uncertainty of this prediction. Thus, models instantiated further down the chain produce a higher uncertainty in their predictions.

As all causal models in OpenAERA can be used for forward and backward chaining, the same must hold for probabilistic models. Given the fact that the modeled function can be used in both directions, it is implicit that an inverse of the function exists. Each causal model includes a noiseless backward function

$$\mathbf{x}_{k-1} = \mathbf{B}\mathbf{x}_k - \mathbf{C}\mathbf{u}_{k-1} \tag{3}$$

with $\mathbf{B}$ representing the inverse function of $\mathbf{F}$ ($\mathbf{F}^{-1}$ in most cases). The backward propagation for any probabilistic model therefore becomes

$$\mathbf{P}_{k-1} = \mathbf{B}\mathbf{P}_k\mathbf{B}^T - \mathbf{C}\mathbf{P}_{control}\mathbf{C}^T - \mathbf{Q}_k \tag{4}$$

This means that the maximum uncertainty of a goal-leading state can be calculated, providing information about the necessary precision of interventions.

Process noise provides another opportunity to optimize causal reasoning systems using uncertainty. While there exists a trade-off in most robotics applications when choosing the process noise, it can be useful when applied in a learning system. When, for example, looking at Kalman Filters, it is important for the designers to choose an appropriate process noise. Too low process noise can lead to the filter ignoring rapid deviations from the expected outcome, and too high process noise makes the filter too sensitive to noisy environments.

In the case of OpenAERA, we can assume low process noise and see deviations from the expected outcome as faulty causal models. Expected and observed outcomes should only diverge rapidly if the instantiated model does not reflect the dynamics of the observed system. This information can be used to generate new hypotheses about the true dynamics, leading to new models that describe the system better. We, therefore, neglect the estimation of process noise in this work and will extend it to include the identification of described changes to causal models in the future.

4 Reasoning algorithm

In the following section, we give a deeper insight into the reasoning algorithm used in OpenAERA and show how uncertainty propagation can be included. We focus on the forward and backward chaining processes used in the planning of control sequences leading to the goal. Backward chaining (abductive reasoning) is used to constrain the search space to relevant models. Forward chaining (deductive reasoning) produces an executable series of commands, representing a plan

of interventions to reach the goal state $\mathbf{g}$. Both backward and forward chaining - excluding the uncertainty propagation - is already implemented in OpenAERA.

Input:

- Current set of observed variables $V_{observable}$ at time t_0 and their values describing the state $\mathbf{x}_0$ and their uncertainty $\mathbf{P}_0$.
- A goal described as a sub-state of all observable variables such that $V_{goal} \subseteq V_{observable}$ with a certain value assigned to them at a certain time t_g such that $\mathbf{x}_{t_g} = \mathbf{g}$ with a maximum uncertainty $\mathbf{P}_{t_g} = \mathbf{P}_g$
- The set of requirement and anti-requirement models M_{req}, as well as the set of causal command and reuse models M_{causal} known by the system.
- The set of probabilistic models M_{prob} which accompany the causal models.

Backward chaining is the depth-first search of possible paths from the goal to the current state using causal models and their requirements. Backward chaining goes back through time, starting at the time at which the goal is supposed to be reached t_g and stepping backward until the current time t_0 is reached.

1. Create a new, empty set of goal requirements G_{req} to be filled.
2. Create a new, empty set of currently instantiable causal models M_{causal,t_0}
3. For each requirement model $m_{req} \in M_{req}$:
   - If m_{req} can be instantiated with the current set of observations - i.e., all left-hand-side (LHS) variables can be bound to currently observable variables and fulfill all conditions of m_{req}:
     Add the instantiation of the causal model on the right-hand-side (RHS) of m_{req} to the set of currently instantiable causal models M_{causal,t_0}
4. Identify all causal models whose set of right-hand-side variables V_{RHS} overlaps that of the set of goal variables V_{goal} such that $V_{RHS} \cap V_{goal} \neq \emptyset$ and create a set from the identified model M'_{causal} with $M'_{causal} \subseteq M_{causal}$.
5. For each model $m_{causal} \in M'_{causal}$:
   (a) If $m_{causal} \in M_{causal,t_0}$:
   Continue loop.
   (b) Bind all variables of m_{causal} and its accompanying probabilistic model $m_{prob} \in M_{prob}$ that are part of g to the value of that variable in g. Leave other variables unbound to be filled during forward chaining.
   (c) Make the instantiation of m_{causal} under m_{prob} with the bound variables a goal requirement g_{req} and add it to G_{req}.
   (d) Identify all requirement models which have the instantiation of m_{causal} on their RHS, creating a subset $M'_{req} \subseteq M_{req}$.
   (e) For each requirement model $m_{req} \in M'_{req}$:
      i. Make instantiating the LHS of m_{req} a sub-goal g_{sub} with the uncertainty of the accompanying probabilistic model m_{prob} as $\mathbf{P}_{g_{sub}}$.
      ii. Set G to g_{sub} and start recursion from 4.
6. **Return** the set of bounded goal requirements G_{req}.

Forward chaining provides the deductive reasoning process in which a series of commands is identified that leads to the fulfillment of the goal requirements generated during backward chaining. Forward chaining starts at time t_0 ("now") and moves forward through time, generating predictions of outcomes of causal models.

1. Create a new, empty set of control vectors U to be executed during the task performance, which will be filled during forward chaining.
2. Set the current set of observations as the input I to the system.
3. Create a new set of models M'_{req} of all requirement models that can be instantiated with I by identifying all models whose LHS variable values' likelihood given the observations' uncertainty in I is higher than a threshold.
4. For each requirement model $m_{req} \in M'_{req}$:
    1. Check if the instantiation of the RHS causal model m_{causal} of m_{req} matches one of the goal requirements identified in backward chaining.
    2. If $m_{causal} \in G_{req}$:
        1. Use the backward function of m_{causal} to calculate the control input necessary to transition the state from the LHS to the desired RHS by binding all variables of the LHS ($\mathbf{x}_{k-1}$) to the values in I and the variables of the RHS ($\mathbf{x}_k$) to the values of the goal requirement: $\mathbf{C}\mathbf{u}_{k-1} = \mathbf{B}\mathbf{x}_k - \mathbf{x}_{k-1}$
        2. If m_{causal} can be instantiated given I and the control input:
            1. Apply the forward function of m_{causal} to generate a prediction of the state change: $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{C}\mathbf{u}_{k-1}$
            2. Apply the forward function of the accompanying probabilistic model to calculate an expected uncertainty after the state change: $\mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{C}\mathbf{P}_{control}\mathbf{C}^T + \mathbf{Q}_k$
            3. Check whether the uncertainty of goal-related variables in the calculated $\mathbf{P}_k$ is smaller than the maximum uncertainty defined in the goal requirement $\mathbf{P}_{g,req}$.
            4. If the expected uncertainty is larger: Plan intermediate observations by reducing the magnitude and duration of $\mathbf{u}_{k-1}$ thus reducing the actuator noise described by $\mathbf{C}\mathbf{P}_{control}\mathbf{C}^T$ and allowing for more observations during command execution.
            5. Set the RHS of m_{causal} as a new predicted observation in I.
            6. Add the control $\mathbf{u}_{k-1}$ to U.
5. **Return** U.

The generated plan thus consists of a set of commands U, each assigned to a certain time period and given a set of input variables that must match observations at this time to perform the control with the expected outcome.

Anti-requirement models: A special focus must be put on anti-requirement models during the reasoning process. Anti-requirement models constrain the solution space of the task by describing states under which a causal model may

not be applied. (The evaluation of anti-requirement models was left out in the previous description of the algorithm for readability.) Anti-requirement models play a role in both backward and forward chaining: In backward chaining, when identifying relevant requirement models (step 5d), anti-requirement models are identified as well, and a goal requirement to *not* instantiate the LHS of the anti-requirement model is generated. In forward chaining, these anti-goal-requirements are in turn evaluated. If instantiating a causal model produces variables that are part of an anti-requirement, the likelihood of the instantiation of the anti-requirement model, given the produced uncertainty of the prediction, is calculated. If this likelihood is over a given threshold, there are three options: (1) The system can choose an alternative path, if available; (2) The magnitude and duration of control inputs can be reduced to minimize the probability of instantiating the anti-requirement model; or (3) abort the current plan and redo the abductive backward chaining process with the assumption that the model chain in question will not lead to the goal.

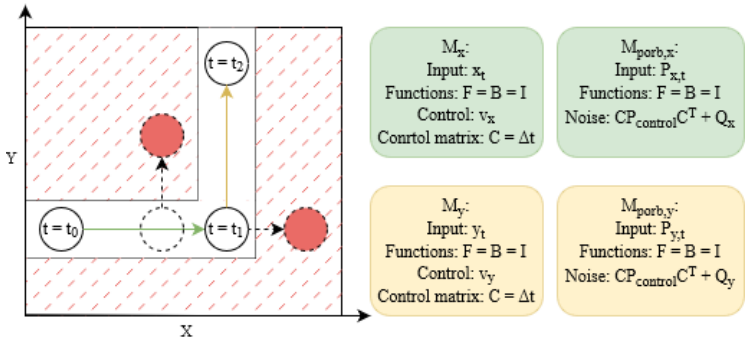

Fig. 1. Task of moving along a constraint space. Left: Visualization of the task with initial state at time t_0, goal state at time t_3. Red areas: forbidden areas where the system may not move; white circles: initial position at time t_0 and after applying the models M_x at time t_0 and M_y at time t_1; red circles: examples of failures of the task if the action of M_x is executed imprecisely. Right: Causal models M_x and M_y and their accompanying probabilistic models $M_{prob,x}$ and $M_{prob,y}$, respectively.

Figure 1 shows a very simplistic task of moving along a constrained space (e.g., a robot moving in a constraint work area). As can be seen, the imprecision of executing model M_x can lead to task failure. The system can identify this during forward chaining, decrease the time during which the command of M_x is executed and thus reduce the impact of actuator imprecision. Executing the command repeatedly with intermediate observations to adjust the duration of the next command can overcome possible failure states.

5 Discussion and future work

We have presented a new approach that extends causal reasoning to address erroneous noisy data in the input and output stream of a controller, and shown

how this is to be implemented in OpenAERA. The resulting reasoning process can better predict possible outcomes of planned interventions to adjust its plans.

The limitation to normal-distributed data can be lifted by changing the uncertainty estimation and propagation process. For example, by using neural networks to estimate the probability distributions given observations and interventions. However, it remains future work how the full reflectability and explainability of AERA in such approaches.

Aside from the application to noisy data, this approach can further be extended to use divergences between predictions and actual observations to enhance the causal discovery process. Detected outliers imply erroneous causal models, which can be corrected through self-reflection mechanisms in AERA.

References

1. Leonard M. Eberding. Comparison of machine learners on an ABA experiment format of the Cart-Pole Task. In *International Workshop on Self-Supervised Learning*, pages 49–63. PMLR, 2022.
2. Leonard M Eberding, Kristinn R. Thórisson, Arash Sheikhlar, and Sindri P. Andrason. Sage: Task-environment platform for evaluating a broad range of AI learners. In *Artificial General Intelligence: 13th International Conference, AGI 2020, St. Petersburg, Russia, September 16–19, 2020, Proceedings 13*, pages 72–82. Springer, 2020.
3. Ben Goertzel, Matthew Iklé, Izabela Freire Goertzel, and Ari Heljakka. *Probabilistic logic networks: A comprehensive framework for uncertain inference*. Springer Science & Business Media, 2008.
4. Patrick Hammer and Tony Lofthouse. 'OpenNARS for applications': Architecture and control. In *Proc. Artificial General Intelligence: 13th International Conference, AGI 2020, St. Petersburg, Russia, September 16–19*, pages 193–204. Springer, 2020.
5. Eric Nivel, Kristinn R. Thórisson, Bas R. Steunebrink, Haris Dindo, Giovanni Pezzulo, Manuel Rodriguez, Carlos Hernández, Dimitri Ognibene, Jürgen Schmidhuber, Ricardo Sanz, et al. Bounded recursive self-improvement. *arXiv preprint arXiv:1312.6764*, 2013.
6. Arash Sheikhlar, Leonard M. Eberding, and Kristinn R. Thórisson. Causal generalization in autonomous learning controllers. In *Artificial General Intelligence: 14th International Conference, AGI 2021, Palo Alto, CA, USA, October 15–18, 2021, Proceedings 14*, pages 228–238. Springer, 2022.
7. Kristinn R. Thórisson. A new constructivist AI: From manual methods to self-constructive systems. In *Theoretical Foundations of Artificial General Intelligence*, pages 145–171. Springer, 2012.
8. Kristinn R. Thórisson. Machines with autonomy & general intelligence: Which methodology? In *Proceedings of the Workshop on Architectures for Generality and Autonomy*, 2017.
9. Kristinn R. Thórisson. The 'explanation hypothesis' in general self-supervised learning. In *Proceedings of Machine Learning Research, 159*, pages 5–27, 2021.
10. Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.
11. Pei Wang. *Non-Axiomatic Reasoning System: Exploring the essence of intelligence*. Indiana University, 1995.
12. Pei Wang. *Rigid flexibility: The logic of intelligence*. Springer Science, vol. 34, 2006.