Evaluating Model Abstractions: A Quantitative Approach

Hessam S. Sarjoughian, Bernard P. Zeigler, and François E. Cellier

Dept. of Electrical & Computer Engineering
University of Arizona
Tucson, AZ 85721-0104

ABSTRACT

An "evaluation" approach devised for an inductive reasoning system called Logic-based Discrete-event Inductive Reasoner is the focus of this paper. The underlying inductive reasoning methodology utilizes abstractions as its primary means to deal with lack of knowledge. Based on abstractions and their treatments as assumptions, the Logic-based Discrete-event Inductive Reasoning system allows non-monotonic predictions. The evaluation approach takes into account explicitly the role of *abstractions* employed in non-monotonically derived multiple predictions. These predictions are ranked according to the *type* and *number* of abstractions used. The proposed evaluation approach is also discussed in relation to the dichotomy of model validation and simulation correctness.

Keywords: Abstraction, Artificial Intelligence, Inductive Reasoning, Model/Simulation Evaluation, Model Validation, Non-monotonic Reasoning, Simulation Correctness.

1. INTRODUCTION

Modeling and simulation practioners have a genuine need to understand the artifacts of their discipline — namely models and simulations. Every useful model and/or simulation must satisfy some objectives — e.g., does the model represent the part of the reality we are interested in? Indeed, modeling and simulation (M&S) methodologies and more widely M&S tools provide some means by which models and simulations can be evaluated.

Due to the underlying fundamental differences in modeling and simulation methodologies (e.g., deductive vs. inductive), various evaluation techniques have been proposed and implemented. Furthermore, within each category of modeling and simulation such as deductive reasoning, there exists no consensus on any "single" evaluation approach to be universally applicable. Indeed, given the many facets of modeling and simulation, it is necessary to devise *model validation* and *simulation correctness* techniques for each M&S methodology. For example, in deductive reasoning, various techniques are employed for evaluating the validity of models represented in it and consequently the correctness of their simulations.[1–5]

2. INDUCTIVE REASONING

In Systems Theory, a system is represented in terms of its inputs, outputs, states, state transition and output functions.[6,7] We refer collectively to states and functions of a system as its *structure*. Based on inputs, outputs, and structure of a system, three distinct reasoning approaches have been identified. Depending on which two of the three are known and which other one is to be predicted, a reasoning mechanism is called *abductive*, *deductive* or *inductive*.

When the output is to be predicted given input and structure, then the reasoning method is referred to as deductive. Abductive reasoning is concerned with determining the input set as opposed to the output set. Deductive and abductive reasoning generally have a "closed-form" solution with the implication that once the structure of a system is identified, it is applicable to all inputs. In inductive reasoning, we call inputs and outputs of a system its *data set*. From this vantage point, the goal of an inductive reasoning approach is to provide a way to obtain a system's structure given some relevant data — a data set. In inductive reasoning, since only a subset of infinitely many possible input and output pairs may be available, a system's structure can only be *partially* determined for a finite set of input/output pairs. An input and output form a "pair" if the former causes the latter.

Further author information -
H.S.S. (correspondence): Email: Hessam@ECE.Arizona.Edu; Ph: 520-626-4846; Fax: 520-621-8076 WWW: http://www-ais.ece.arizona.edu/~hessam/
B.P.Z.: Email: Zeigler@ece.arizona.Edu; WWW: http://www-ais.ece.arizona.edu/~zeigler/
F.E.C.: Email: Cellier@ECE.Arizona.Edu; WWW: http://www.ece.arizona.edu/~cellier/

Given the distinct characteristics of inductive reasoning as opposed to deductive (abductive) reasoning, the evaluation of a system's behavior requires distinct treatment depending on the particular reasoning mechanism. We continue with a brief description of the inductive reasoning methodology underlying the evaluation approach being discussed here.

2.1. Inductive Prediction

Inductive reasoning has to rely on incomplete data set.[8,2,3] Therefore to overcome the lack of complete data, it is necessary to extend the data set in a way to allow making predictions. A compelling strategy is to augment the data set with *hypothesized data* based on some "assumptions.[9] " These assumptions provide the basis to *derive* the unavailable data. However, since assumptions are used in obtaining the hypothesized data, the derivation is said to be *non-monotonic.* The word non-monotonic is used to indicate that the derived data holds as long as its underlying assumptions hold. Once knowledge contrary to the supporting assumptions becomes available, the hypothesized data must be discarded.

The enabling means for inductive (or non-monotonic) prediction are assumptions that directly relate to abstractions. We use the term abstraction to refer to "generalization" of knowledge. Suppose, we have two input segments: ω_1 and $\omega_1^{'}$ where the latter is an abstraction of the former. An input segment ω_1 can be specified in terms of its input event, duration, and state. Using abstraction, an input segment may be represented and used in terms of either its input event or state. Two abstraction types for an input segment are *input* and *state*. For example, we can make an abstraction $\omega_1^{'}$ from ω_1 that has associated with it an assumption (e.g., input abstraction.)

Based on the Iterative Input/Output Function Observation (IOFO) specification (see Section 2.1) and non-monotonic reasoning, an inductive reasoning approach called Logic-based Discrete-event Inductive Reasoning (LDIR) has been proposed to make predictions in the absence of complete knowledge.[9,10] An implementation of the proposed approach has been developed using a Logic-based Truth Maintenance System (LTMS).[11] The LDIR stores two types of input/output segment pairs: *observed* and *hypothesized.* The observed input/output segments are recorded as "assertions." The remaining input/output segments are recorded as "hypotheses" and are associated with their assumptions.

2.2. Iterative IO Function Observation Structure

In this section, we briefly discuss a mathematical structure that represents a system in terms of its data set and structure.[9] This structure underlies the Logic-based Discrete-event Inductive Reasoner. A system's input/output trajectories (i.e., $IOspace = \{(\omega, \psi) \mid (\omega, \psi) \in (X, T) \times (Y, T), dom(\omega) = dom(\psi).\}$) can be partitioned into input/output generator segment pairs — (Ω_G, Ψ_G). By partitioning input/output trajectories into input/output generator segments, the *IOspace* can be constructed by concatenation. In this way not only the original input/output trajectories can be recreated, but also others. The collection of all input/output generator segments of a system is defined as

$$\begin{aligned} IOspace_G = \quad & \{(\omega_G, \psi_G) \mid (\omega_G, \psi_G) \in IOspace, \\ & \omega_G \in \Omega_G, \psi_G \in \Psi_G, \\ & IOspace_G \subseteq IOspace, \\ & dom(\omega_G) = dom(\psi_G)\} \end{aligned}$$

Each input segment has an input event and a duration. Similarly, the output segment has an output event and a duration. In order to represent a system explicitly in terms of its states, we associate input and output segments with states. In LDIR, an input/output generator segment is represented as $((s_i, (x_{val}, dt)), (s_f, (y_{val}, dt)))$ where initial and final states are denoted as s_i and s_f.

For a given system, if the set of all IO generator segments are available, then the system's input/output trajectories of arbitrary length can be constructed. Given $IOspace_G$, an *Iterative Input Output Function Observation* specification for a causal, time-invariant IO function observation structure can be defined as:

$$G_F = \langle T, X, S_i, Y, IOspace_G, F_G, \gamma_G \rangle$$

where

T	time base
X	input value set
Y	output value set
S_i	set of initial states
$IOspace_G$	causal, time-invariant input/output segment generator set
F_G	IO function generator set
γ_G	final state hypothesizer

with the following constraints:

$$F_G : S_i \longrightarrow \text{ partial } IOspace_G ,$$
$$\gamma_G : S_i \times IOspace_G \longrightarrow S_i .$$

The *free* Iterative IOFO specification allows nonempty finite concatenations of elements of $IOspace_G$ as well as F_G. That is, predicted input/output trajectories can be composed of generator input/output segments. Hence, given a data set containing I/O segments and an input trajectory, the goal is to predict an output trajectory. However, if F_G does not represent a system's complete IO behavior, then it may be necessary to *compose* trajectories from segments, some of which have to be predicted based on some belief set that we denote as an *assumption set*. As a case in point, consider the composition of two segments ω_1 and ω_2 ($\omega_2 \circ \omega_1$.) However, unless the final state of ω_2 is the same as the initial state of the ω_1, we have no choice but to make the composite trajectory a hypothesis. By ignoring a mismatch between the final state of ω_2 and the initial state of the ω_1, we would have a hypothesized trajectory. That is, we need to make either an abstraction of ω_1 ($\omega_1^{'}$) or ω_2 ($\omega_2^{'}$) in order to make a composite trajectory such as $\omega_2^{'} \circ \omega_1$ or $\omega_2 \circ \omega_1^{'}$.

An input trajectory may be partitioned into segments based on the duration of the input segments that are in a data set. Each partitioned input segment can have a duration equal to an input segment in the data set. Also, the duration of a partitioned input segment can be equal to the duration of an input segment having the longest duration. Similarly, a partitioned input segment's duration can be neither exact nor longest (i.e., somewhere between exact and longest.) We call matching of an input segment's length against one in the data set as either *exact*, *longest*, or *all*. For example, if we specify an "exact" match, then the input trajectory will be partitioned as to match the data set input segments that have the exact duration as the partitioned input segment. That is, an assumption set can be identified to have elements from $\{longest, exact, all\}$ and $\{input, state\}$. Hence, the G_F can be reformulated as an assumption-based iterative IOFO structure using assumption sets.

2.3. Example

We describe an example for the remainder of this paper. We suppose there exists a shipyard that has two Repair Stations called RS-1 and RS-2 where the former adheres to a First-In-First-Out (FIFO) discipline and the latter to a Priority Ranking (PR) discipline.[9] The shipyard receives different types of vessels in need of repair with different priorities. For example, vessels of type "A" have the highest priority of all and vessels of type "B" are given priority over those of type "C."

Only one of the two repair stations is assumed to be in operation for any given period. As a scenario, we suppose vessels in need of repair enter the shipyard at stochastically chosen time points, wait for their turn, and depart the shipyard after being repaired. Each vessel type has an assigned I.D. number indicating how much time is required for it to be fixed. In this simple example, there are three types of vessels — "A," "B," and "C" — that require one unit, two units, and three units of time, respectively, for repair. We exclude the condition where multiple vessels can be repaired in parallel.

An observer may keep an ordered record of the vessels entering the repair station, while removing from the list any vessel departing from it. In order to completely specify the state of a repair station, it is necessary not only to have an ordered record of the vessels entering the repair station, but also to specify how much time is necessary for each vessel until it is able to depart again, given that some vessels are already waiting for repair. The "state" of each repair station can be described by:

$$(\cdots,\ (id_j, t_j),\ \cdots)$$

where id_j denotes the identity of the vessel "J," and t_j denotes the remaining time before vessel "J" will depart from the repair station. We choose the input/output trajectories for the shipyard to have as their states $(num\_of\_vessels,\ (\cdots,\ id_j, \cdots))$ to make the problem non-trivial. Even though $num\_of\_vessels$ can be computed from $(\cdots,\ id_j, \cdots)$, for convenience, we make this information explicit. Each IO trajectory has as its input events the identities of arriving vessels. Likewise, the identities of leaving vessels are the output events of an IO trajectory.

We can represent an observed input/output trajectory (IO Trajectory I) for the shipyard example as:

```
io-traj-1:  ((si-0 ()) (in a 1) (out a 1))
            ((si-0 ()) (in nil 1) (out nil 1))
            ((si-0 ()) (in b 2) (out b 2))
            ((si-0 ()) (in nil 1) (out nil 1))
            ((si-0 ()) (in c 3) (out c 3))
            ((si-0 ()) (in nil 1) (out nil 1)).
```

An input trajectory such as *io-traj-1* is simply a list of records, each containing an initial state, an input segment, and an output segment, in the order given. The final state associated with each record is the initial state of the record following it, except for the last one. The arrival or departure of no vessel at either the beginning or the end of a segment is indicated as `nil`. For instance, `(in nil 1)` specifies that no new vessel arrived for this segment. (Figure 1 depicts the above trajectory and IO Trajectory II which satisfies a FIFO discipline only. The second IO trajectory, which satisfies a FIFO discipline only, is shown in the bottom graph of Figure 1. Its pseudo-code representation is:

```
io-traj-2:  ((si-0 ()) (in b 1) (out nil 1))
            ((si-1 (b)) (in a 1) (out b 1))
            ((si-1 (a)) (in nil 1) (out a 1))
            ((si-0 ()) (in nil 3) (out nil 3))
            ((si-0 ()) (in a 1) (out a 1))
            ((si-0 ()) (in nil 1) (out nil 1))
            ((si-0 ()) (in c 2) (out nil 2))
            ((si-1 (c)) (in b 1) (out c 1))
            ((si-1 (b)) (in nil 2) (out b 2))
            ((si-0 ()) (in nil 1) (out nil 1)).
```

Consequently, the first record of *io-traj-1* says that there were no vessels in the repair station initially, and that a vessel with identity `a` arrived at that time, which departed again one time unit later. The second record, `((si-0 ()) (in nil 1) (out nil 1))`, says that no vessel arrived or departed during the second time unit and that no vessels were in the repair station during this time period, etc.

Let us suppose the following input trajectory for which LDIR is supposed to predict its output trajectory with a given initial state and an assumption set:

```
in-traj-3:  (in c 0)
            (in a 1)
            (in b 6))
            (in nil 8))
```

Since we know the internal structure of the system, we can of course perform a quantitative and completely deductive reasoning, to determine the true IO trajectory for this system, given `in-traj-3` as its input trajectory. Note that we use the knowledge about the shipyard's internal structure as a baseline to measure the predictions of LDIR against it.

We assume that the shipyard is initially empty and that it follows a FIFO discipline. The pseudo-code representation of the correct IO trajectory is as follows:

```
io-traj-3-fifo:  ((si-0 ()) (in c 1) (out nil 1))
                 ((si-1 (c)) (in a 2) (out c 2))
                 ((si-1 (a)) (in nil 1) (out a 1))
                 ((si-0 ()) (in nil 3) (out nil 3))
                 ((si-0 ()) (in b 2) (out b 2))
                 ((si-0 ()) (in nil 1) (out nil 1))
```

Using LDIR, we can predict a set of logically consistent IO trajectories for this input trajectory assuming no other knowledge about the system except for *io-traj-1* and *io-traj-2*. The predicted output trajectory for *in-traj-3*, initial state `(si-0 ())`, and assumption set (`exact abs-input`) is shown in Figure 1. The assumption set specifies that a candidate input segment as a result of partitioning must have equal duration to one available in the data set. Furthermore, it specifies that the input event can be excluded (abstracted) in finding a match for the candidate input segment. In Section 3.1, we assess the fidelity of these hypothesized IO trajectories relative to the correct one.

3. QUANTITATIVE EVALUATION OF PREDICTED IO TRAJECTORIES

Obviously, the predictions made by the Logic-based Discrete-event Inductive Reasoner (LDIR) would need to be evaluated, especially when multiple predictions are possible. As we indicated earlier, LDIR interacts with a data set containing observed and hypothesized I/O segments. Therefore, the evaluation of the LDIR would have to be carried out in terms of its data set maintained by the LTMS. We can assess the quality of the database maintained by LTMS, and the effectiveness of LDIR in using the available knowledge for predicting IO trajectories. We can say that the database maintained by LTMS constitutes the *model* of the system, whereas predicting an IO trajectory represents a qualitative *simulation*. Loosely speaking, evaluating the quality of the database is synonymous with *validating the model*, whereas evaluating the quality of a prediction corresponds to *verifying the simulation*.

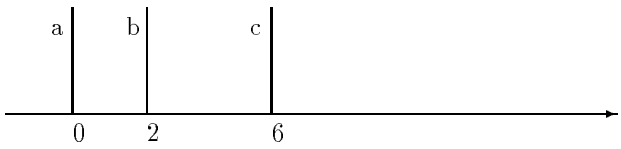
More traditional modeling/simulation systems use goodness-of-fit measures to assess the quality of a simulation. Also, since the model usually remains static during the simulation, it is possible to separate the task of validating the model from that of verifying the simulation. However, in LDIR, and in other qualitative modeling/simulation systems (e.g., SAPS[12]), these tasks are more intricate. In LDIR, the model does not remain static during the simulation (new hypotheses are added on the fly, and the available knowledge is constantly revised to maintain consistency among all available facts and hypotheses).

Before we can define precisely what is the quality of the LTMS database (the model), we need to describe what the term "quality" entails. The quality of a knowledge maintenance system can be specified in terms of its *reliability* and *completeness*. The following illustrates these two properties.

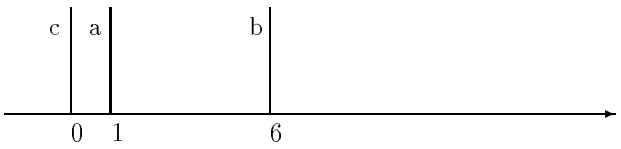
The warehouse of a large company stores many parts that it must sell when they are needed. The warehouse manager keeps a book in which all parts are listed with their part numbers, location in the warehouse, and prices. Unfortunately, the knowledge is incomplete. There are many parts for which no prices are listed, and other parts are missing altogether. The warehouse manager now has two choices. He can work with the book as is. In this case, the *reliability* of the available knowledge is excellent, yet the *completeness* is not. If the apprentice is to sell a part for which no price is listed, he must always look for the warehouse manager to find out how much to charge. The other possibility is for the apprentice to estimate the cost of the incomplete entries as best as the manager can within the limitations of available time. As missing entries are added to the book, the knowledge becomes more *complete*. Unfortunately, the *reliability* will suffer in the process.

The same is true for any database maintained by the LTMS. The more hypotheses are added to the database, the more complete the knowledge becomes. However, the increase in completeness goes hand-in-hand with a reduction in reliability. Thus, when evaluating the quality of the database, we shall define two separate quality measures — one measuring the reliability of the database, and the other measuring its completeness.

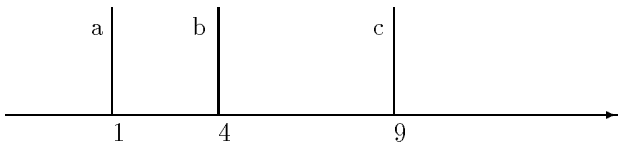
The completeness of an LTMS data set may be specified, in part, by the *cardinality* of the spaces of hypothesized IO segments. For a given set of input segments, hypothesized IO segments can be those that are non-monotonically derived based on some observed IO segments and an assumption set. For example, using abstraction `abs-length`



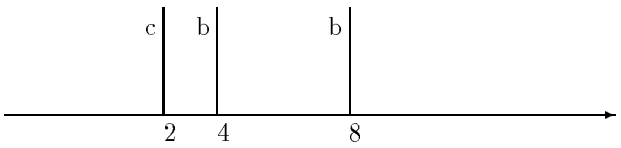
Observed Input Trajectory 1



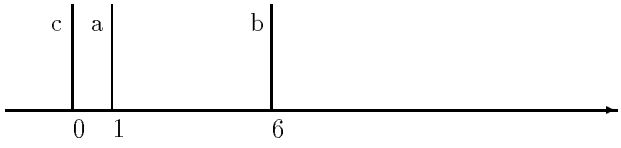
Candidate Input Trajectory I



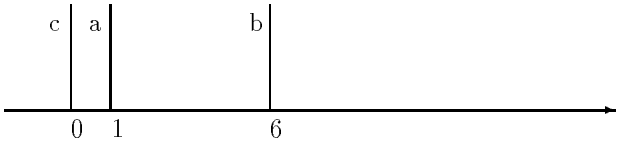
Observed Output Trajectory 1



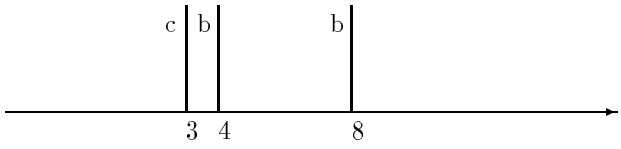
Predicted Output Trajectory I



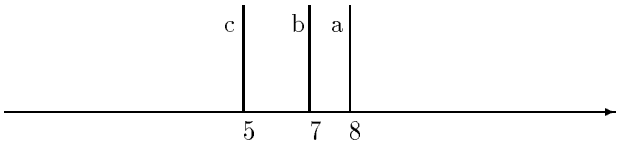
Candidate Input Trajectory 3



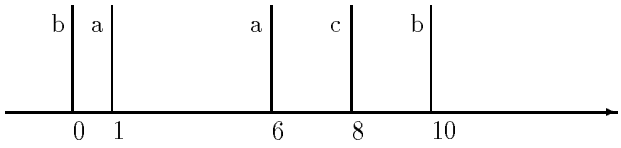
Candidate Input Trajectory II



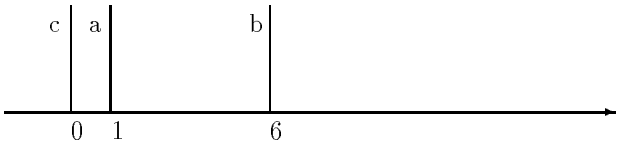
Correct Output Trajectory 3



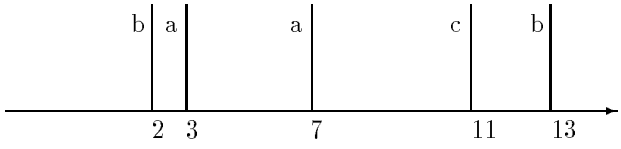
Predicted Output Trajectory II



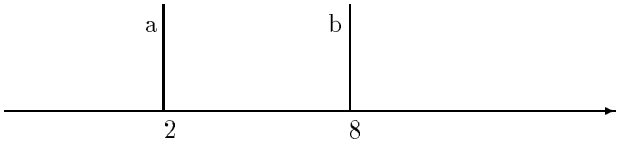
Observed Input Trajectory 2



Candidate Input Trajectory III



Observed Output Trajectory 2



Predicted Output Trajectory III

Figure 1. FIFO Discipline Example

with some observed IO segments leads to a space of hypothesized IO segments for a set of input segments. Other spaces of hypothesized IO segments for the same set of input segments and observed IO segments can be non-monotonically derived using `abs-input`, `abs-state`, and their combinations (e.g., `(abs-length, abs-input)`.) Each of the abstraction types usually leads to different hypothesized IO segments. Hence, the completeness measure is related to the cardinality of observed and hypothesizable IO segments.

The reliability of hypothesized IO segments is related to the type of abstraction employed since each abstraction enforces a certain degree of *predictability*. We can rank the three types of abstraction in order of increasing *compliance* as: (1) length, (2) input event, and (3) state. We are using the term "compliance" to indicate how forgiving an abstraction is. The more compliant an abstraction is, the less reliable a hypothesis will be, and the smaller the fidelity of a predicted IO segment that is based on this hypothesis. State abstractions impose much stronger compliance in generating hypothesized IO segments than either input event or length abstractions. Having the abstraction defined as given above, it is straightforward to order their combinations as well. If length, input event, and state are abstracted simultaneously, then the so hypothesized IO segments are based on the most compliant assumptions, allowing prediction of *anything* as long as consistency is not violated!

The compliance of an abstraction type offsets the *fidelity* of IO segments hypothesized based on it. In the most restrictive sense, the term "fidelity" might refer to whether a hypothesized IO segment will eventually be confirmed or rejected. However, instead of postulating a dichotomy, we use the term fidelity to indicate to what extent a hypothesized IO segment is consistent with the available IO segments. The stronger the compliance of an abstraction, the worse the fidelity of hypothesized IO segments generated based on it — the fidelity of IO segments degrades as abstractions with higher compliance are used. Thus, we use the terms compliance and fidelity in relation to each other.

How does the predictability of LDIR relate to the number of observed IO segments? The answer to this question depends on the candidate input trajectory. The more useful IO segments have been observed for a system in relation to a candidate input trajectory, the fewer IO segments would need to be hypothesized. Also, more observed IO segments place more restrictions on what IO segments can be hypothesized.

Hence the size of the space of the hypothesized IO segments is related to both the number of observed IO segments and the types of abstractions used. The cardinality of the space of hypothesizable IO segments based on `(abs-length, abs-state)`, `(abs-length, abs-input)`, or `(abs-length)` decreases in the order given. Furthermore, while the number of hypothesizable IO segments decreases as less compliant types of abstractions are employed, their fidelity improves.

Having different types of IO segments (some asserted, others hypothesized), it is possible to define two heuristic quality measures relating to: (1) the quality of all available IO segments in the database, i.e., the quality of the model, and (2) the quality of a predicted IO trajectory given an input trajectory, i.e., the quality of a simulation. A *quality measure* is a real-valued number in the range 0.0 to 1.0, where larger values denote improved quality.[3]

We introduce the term *IO segment fidelity* in terms of a *Figure of Merit (FoM)* assigned to it. Let F_{abs_i} denote the absolute fidelity of the i^{th} IO segment. Asserted IO segments have the highest figure of merit, whereas hypothesized IO segments using the abstraction mechanism with the largest compliance have the lowest figure of merit. Here, we consider the possibilities from the first two elements of `elm-1` $\in$ {`longest, exact, all`} together with `elm-2` $\in$ {`abs-input, abs-state`} with some (arbitrarily) assigned figures of merit for each abstraction (assumption) type. Table 1 suggests some assigned FoM for each combination of abstraction types. In the chosen scale, the maximum and minimum absolute fidelity values are $F_{\max} = 4.0$ and $F_{\min} = 0.0$, respectively.

Now, the *relative fidelity* of each IO segment can be defined as:

$$F_{\text{rel}_i} = \frac{F_{\text{abs}_i}}{F_{\max}}$$

Assertions have the highest relative fidelity (1.0), whereas hypotheses that are based on the abstractions with the highest compliance have the lowest relative fidelity (0.0). Given all currently available IO segments in the LTMS database, we can define the *average relative fidelity* as:

Assumption used	FoM
—	4.0
`exact`	3.5
`longest`	3.0
`(exact, abs-input)`	2.5
`(longest, abs-input)`	2.0
`(exact, abs-state)`	1.5
`(longest, abs-state)`	1.0
`(exact, abs-input, abs-state)`	0.5
`(longest, abs-input, abs-state)`	0.0

Table 1. Figures of merit assigned to some types of assumptions.

$$F_{\text{avg}} = \frac{1}{n} \cdot \sum_{i=1}^{n} F_{\text{rel}_i}$$

where n denotes the number of IO segments currently stored in the database. F_{avg} can also be used as a quality measure.

Given that we have a finite set of input segments, each either being a hypothesis or an assertion, we call the number of all possible combinations of initial states and input events N_p. Similarly, we call the set of all combinations of different initial states and input events that are currently present among the (asserted and hypothesized) IO segments N_a, where $N_a \leq N_p$.

We can introduce the *evidence ratio measure*:

$$E_R = \frac{N_a}{N_p}, \qquad 0.0 < E_R \leq 1.0$$

We shall use this measure as the *completeness measure* of our model. The interpretation of E_R is that, the greater the value of E_R, the fewer IO segments will need to be hypothesized in relation to a fixed set of initial states and input events. Smaller values of E_R indicate the converse. A value of $E_R = 1.0$ indicates that LDIR will never have to resort to either input or state abstractions in hypothesizing IO segments. We have not included length abstractions since they generally do not contribute greatly to the evidence ratio measure. Now, given F_{avg} and E_R, we can define Q_{Model}, which is the *prediction quality* of the model (i.e., the overall quality of all available IO segments in the LTMS:)

$$Q_{\text{Model}} = F_{\text{avg}} \cdot E_R.$$

The two influencing factors of the prediction quality Q_{Model} are always in competition with each other. Evidently, if the cardinality of all possible IO segments is exhausted through observations, $Q_{\text{Model}} = 1.0$. A useful feature of quality measures, as they were defined in,[3] is that multiple (usually competing) quality measures can be simply multiplied with each other, leading to a multidimensional new quality measure that takes into consideration all the influencing factors assessed through the individual quality measures contributing to it.

Hence, if no hypotheses have been made, F_{avg} shows a perfect score of 1.0. When only few IO segments are observed or hypothesized, E_R will be poor. On the other hand, if everything has been hypothesized that can be, E_R shows a perfect score of 1.0, but this time around, F_{avg} will be poor.

If we don't know yet what we wish to use the model for, i.e., which input trajectory we are going to use to predict an output trajectory, this is the best we can do. However, given an input trajectory, we can evaluate the quality of the IO trajectory predicted by LDIR in more direct ways, i.e., we can predict the *quality of a simulation*.

	F_{avg}	E_R	Q_{Model}	Q_{Simul}
`IO-Trajectory-I(asm-1)` asm-1: {`longest, abs-input`}	0.90	0.63	0.56	0.14
`IO-Trajectory-II(asm-2)` asm-2: {`exact, abs-input`}	0.96	0.56	0.54	0.54
`IO-Trajectory-III(asm-3)` asm-3: {`longest, abs-state`}	0.94	0.56	0.53	0.32

Table 2. Quality measures for predicted Scenarios 1 through 3.

Since we already know the input segments we shall have to work with, the evidence ratio is of no concern any longer. We only deal with the relative fidelities of individual predicted IO segments. Making the supposition of statistical independence of neighboring predicted IO segments, we can postulate the following quality measure:

$$Q_{\text{Simul}} = \prod_{j=1}^{m} F_{\text{rel}_j}$$

where m denotes the number of predicted IO segments. The supposition of statistical independence is obviously a preposterous one. Simulation output is *never* statistically independent (unless we try to predict the next value of a noise generator[13]). However, we don't have anything better to go by, and so we shall have to live with this supposition. Most qualitative simulation systems do. For example, SAPS does exactly the same, except that F_{rel_j} is replaced by a *measure of likelihood* of the prediction made.[14]

3.1. Evaluation of the Shipyard Example

Based on Q_{Model} and Q_{Simul}, we can analyze three predicted output trajectories for the candidate input trajectory shown in Figure 1. We use three assumptions sets — asm-1: {`longest, abs-input`}, asm-2: {`exact, abs-input`}, and asm-3: {`longest, abs-state`}. Based on these, LDIR predicts three scenarios: `IO-Trajectory-I(asm-1)`, `IO-trajectory-II(asm-2)`, and `IO-trajectory-III(asm-3)`.

The evidence ratio can be computed given N_p — that is the set of all combinations of states and inputs. Neither of these are bounded in the shipyard example. Consequently, we need to replace the theoretical cardinality by a much smaller subset: the power set of all initial states and input events that have ever been observed.

Unfortunately (or fortunately), we can't know what we don't know. We have to live with this fact, and make the best of it. LDIR can compute the powerset of all ever observed states. In the case of RS-1, LTMS contains 4 different initial states and 4 input event types. The power set of these two (independent) quantities is 16. Note that we have only a few observed IO trajectories and therefore the tables given below are based on few data points.

The fidelity measure and evidence ratio for repair-station-1 are 1.0 and 0.5, respectively. The perfect score for the fidelity measure is due to having no hypotheses in the data set. Given that there are 4 distinct initial states and 4 input events for RS-1, the total number of combinations of initial states and input segments is 16. The observed IO segments contain 8 different cases with $E_R = 8/16$.

Given the respective predicted output trajectories 1 through 3, the quality measures for each can be computed (refer to Table 3). (The quality measures of the model change since the model itself is updated to include some hypotheses.)

The quality of the model for the IO-trajectory-I indicates that it is best suited for predictions. However, the quality of the predicted trajectory using this model is worse than the other two despite having a superior model quality. The quality of prediction can be adjusted by using a third quality measure, the *assertion ratio measure*:

$$A_R = \frac{N_A}{N_A + N_H}$$

	F_{avg}	E_R	A_R	Q_{Model}	Q_{Simul}
`IO-Trajectory-I(asm-1)` asm-1: {`longest, abs-input`}	0.90	0.63	0.73	0.41	0.14
`IO-Trajectory-II(asm-2)` asm-2: {`exact, abs-input`}	0.96	0.56	0.84	0.46	0.55
`IO-Trajectory-III(asm-3)` asm-3: {`longest, abs-state`}	0.94	0.56	0.84	0.45	0.33

Table 3. Revised quality measures for predicted Scenarios 1 through 3.

where N_A denotes the number of assertions in the database, and N_H represents the number of hypotheses. The modified model quality would then be evaluated as:

$$Q_{\text{Model}} = F_{\text{avg}} \cdot E_R \cdot A_R$$

Using the modified model quality measure, the predicted IO trajectories show simulation quality as expected (see Table 3.) Now, there is a good correspondence between what the model quality stipulates and what the simulation quality confirms. The original definition is left in the text to show that there is a fairly high degree of heuristicism in the detailed definitions of these quality measures, and more fine tuning may be needed down the road; for now, however the modified quality measures look rather promising.

Note that the evidence ratio has gone up in all cases (as it must), yet the fidelity measure and the assertion ratio have both gone down, and the overall model quality has in fact decreased, i.e., we didn't do a very smart thing by augmenting the LTMS with these hypotheses. In all likelihood, at least some of them will have to be revoked (negated) in the future.

Among the three cases, both the model quality measure and the simulation quality measure suggest that Scenario 2 is the best. Scenario 1 added so many spooky hypotheses to the LTMS that its results are the most doubtful ones, although the abstraction mechanism is valued more reliably, in general, than that used by Scenario 3. The model and simulation quality measures agree in their relative assessments of the three scenarios.

Now, we examine each of the predicted scenarios in terms of their behaviors. We begin with `IO-Trajectory-I` (Refer to Figure 1.) The predicted IO trajectory satisfies the FIFO discipline, ignoring incorrect job identity for input event "A". This is accidental however, since inside the shipyard two major "remodeling jobs" have taken place that were not visible from the outside. The amount of time predicted for vessels "C" and "A" to be serviced is incorrect. Moreover, IO segment `(((SI-0 ()) (IN NIL 2)) ((SI-0 ()) (OUT NIL 2)))` is split into two identical IO segments `(((SI-0 ()) (IN NIL 1)) ((SI-0 ()) (OUT NIL 1)))`.[9] This is due to asking for `exact` match and not having `(((SI-0 ()) (IN NIL 2)) ((SI-0 ()) (OUT NIL 2)))` in `IO-space-g`. Evidently, this deviation is harmless. `IO-Trajectory-II` looks better than `IO-Trajectory-I`, but the conclusion may be accidental, since the internally made abstractions are still rather dubious.

How about `IO-Trajectory-II`(Refer to Figure 1?) Evidently, the predicted IO trajectory is not the expected one. Moreover, the order in which vessels are predicted to be repaired is incorrect. Even though the first predicted IO segment is consistent and correct, the remaining part of the predicted IO trajectory is incorrect. LDIR predicted vessel "C" to take longer than expected to be repaired. Consequently, all the remaining IO segments turn out to be incorrect. This, obviously, need not be true in general. The reason vessel "C' requires 5 time units for repair is due to asking for the longest match in the assumption set. Nevertheless, the number of vessels entering and leaving the repair-station-1 is correct.

(`IO-Trajectory-III`) shows the least agreement with the desired predicted IO trajectory. We expected this since the assumption set is `(longest, abs-state)`. The abstraction of initial states is more compliant than that of input events. In this scenario, vessel "C" disappeared altogether!

The above discussion shows that assessing a qualitative simulation in the same way as one would judge a quantitative simulation, i.e., in terms of a goodness-of-fit measure, is problematic at best. The previously presented quality measures are much more solid and reliable, in general, than any goodness-of-fit measure we might come up with.

If this is the case, what is the purpose of the simulation? Had we provided the system with more evidence (a higher evidence ratio) to start with, the quantitative results of the simulation would also have been better. The problem is simply that if the agreement between reality and prediction is poor, as in the shipyard example, this is related to the problem of not having enough evidence, and not to a principal flaw in the methodology. The proposed quality measures have a much better chance of assessing the real strengths and weaknesses of such a model than a simple output-to-output comparison.

However, there is also another implicit benefit of performing logic-based qualitative simulation runs. It relates to the possibility of understanding the underlying reasoning processes of the qualitative simulator. A traditional weakness of simulation is that simulation results are rarely enlightening.[3] It is as difficult to generalize knowledge from a simulation output as from a lab experiment. Many different simulation runs are usually needed until a human researcher can discern the general patterns behind the specific patterns generated by individual simulation runs. In this respect, the LDIR methodology exhibits an important advantage. Its reasoning processes are immediately open to human interpretation. We shall talk more about this facet of the DIR methodology in the following section.

4. OTHER EVALUATION APPROACHES

The average fidelity measure, F_{avg}, and the evidence ratio measure, E_R, are adaptations of the *entropy reduction measure*, H_R, and the *observation ratio measure*, O_R, used in SAPS.[3] SAPS defines the quality of its inductive model in a similar fashion as $Q_M = H_R \cdot O_R$. It uses Q_M to distinguish between the relative virtues of different "masks" (the abstraction mechanism used in SAPS), and chooses as the "optimal mask" the one with the largest Q_M value, i.e., it selects the abstraction mechanism that maximizes the predictability power of the qualitative model. LDIR currently does not need to do so, because there are a small number of choices available in selecting abstraction mechanisms, and their relative merits w.r.t. predictability power are fairly well understood. In the future, however LDIR will be expanded to deal with multi-input systems (as SAPS already does); at such time, Q_{Model} may be used by LDIR as Q_M is currently used in SAPS.

A general evaluation approach for learning systems based on a probabilistic framework has been proposed by Valiant.[15] This approach is devised for systems that learn inductively as opposed to system that learn either by receiving more factual knowledge (learning by being told) or by becoming more efficient while not receiving any new knowledge (speedup learning.) The basic concept for evaluating inductive learning is called *probably approximately correct (PAC)*, that is to say learning unobserved knowledge (e.g., an input/output segment) with high probability. More specifically, PAC is bounded by confidence δ and accuracy ϵ parameters:

$$Pr[error(F, \hat{F}) > \epsilon] < \delta$$

where F is the knowledge to be learned (e.g., observed input/output segments) and $\hat{F}$ is the knowledge that is approximately correct (e.g., hypothesized input/output segments) over some universe of objects U.

In PAC learning, it is assumed that some pieces of knowledge are more important than others. The concept of *approximately correct* captures the degree to which $\hat{F}$ matches F. In LDIR, predictions are ranked based on the type of assumption used. It LDIR, no distinction is made between learning input/output segments that are more important than others. This is due to the presumption that all elements of the LDIR data set (input/output segments) have the same degree of importance. The LDIR learning framework can readily be extended to take into account such knowledge in addition to type of abstractions.

Two extensions of PAC learning are restricted hypothesis bias and preference bias.[16–18] Learning with restricted bias is concerned with the representational syntax. For example, given hypotheses represented in Boolean conjunction in some universe U, the space of possible hypotheses is $2^{|U|}$. Based on the space of hypotheses, ϵ, and δ, bounds on the number of training examples can be obtained. Preference bias learning orders possible hypotheses based on some preference. For example, based on the Occam's Razor principle, hypotheses that have the simplest form would be more favorable. In LDIR, hypotheses are ordered based on the abstractions used (e.g., hypothesized segments having longest duration are less favorable than those having exact duration.)

5. CONCLUSIONS

Given the Logic-based Discrete-event Inductive Reasoning methodology, we discussed an approach that allows evaluation of its predictions. The two major measurement metrics are model and simulation qualities. The abstraction figures of merit (e.g., $FoM_{\{\texttt{exact},\texttt{input}\}}$) were the principal artifacts in obtaining Q_{Model} and Q_{Simul} quality measures. The figures of merit ensured that the ranking of predictions are directly related to the compliance of each abstraction employed. The other contributors to the measurement metrics were the number of observed and hypothesized input/output segments. As it was shown, the quality measures for the shipyard example were as expected in accordance to the abstraction types used.

In terms of future work, it would be necessary to examine the LDIR evaluation approach with sufficiently large-scale examples. That is, the training examples should be statistically valid. The LDIR reasoning mechanism appears to be a suitable learning mechanism to be evaluated using the PAC approach. In particular, for a large data set, it provides a mechanism to obtain predictions having a desired confidence and accuracy measure. However, PAC learning and specifically preference learning, need to be extended to account for the use of abstractions as defined in LDIR. The simple assignment of figure of merits as discussed in this paper may need to be revised to lead to more appropriate model and simulation quality measures.

REFERENCES

1. B.P. Zeigler. *Multi-Facetted Modelling and Simulation*. Academic Press, New York, 1984.
2. M. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.
3. F. E. Cellier. *Continuous System Modeling*. Springer-Verlag, New York, 1991.
4. P.A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds*. Prentince Hall, Inc., Englewood Cliffs, New Jersey, 1995.
5. B.P. Zeigler, H. Praehofer, and T.G. Kim. *Theory of Modeling and Simulation, 2nd Edition*. Academic Press, Inc., New York, 1998.
6. B.P. Zeigler. *Theory of Modeling and Simulation*. John Wiley and Sons, New York, 1976.
7. W.A. Wymore. *Model-based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design*. CRC, Boca Raton, 1993.
8. G.J. Klir. *Architecture of Systems Problem Solver*. Plenum Press, New York, 1985.
9. H.S. Sarjoughian. *Inductive Modeling of Discrete-event Systems: A TMS-based Non-monotonic Reasoning Approach*. PhD thesis, University of Arizona, 1995. Department of Electrical and Computer Engineering.
10. H.S. Sarjoughian and B.P. Zeigler. "Inductive modeling: A framework marrying systems theory and non-monotonic reasoning". *Hybrid Systems II*, LCNS 999:417–435, 1995. Springer Verlag.
11. K.D. Forbus and J. de Kleer. *Building Problem Solvers*. MIT Press, Cambridge, 1993.
12. M. Moorthy, F.E. Cellier, and J.T. LaFrance. Predicting u.s. food demand in the 20th century: A new look at system dynamics. In *SPIE*, April 13-17 1998.
13. A.M. Law and W.D. Kelton. *Simulation modeling and analysis*. McGraw-Hill, New York, 1991.
14. F.E. Cellier, J. López, A. Nebot, and G. Cembrano. "Confidence Measures for Predictions in Fuzzy Inductive Reasonning". *Int. J. General Systems*, In review for a special issue on Fuzzy Inductive Reasoning.
15. L.G. Valiant. "A theory of the learnable". *Communications of ACM*, 27:1134–1142, 1984.
16. A. Blumer et. al. "Occam's razor". *Information Processing Letters*, 24:377–380, 1987.
17. M. Kearns and L.G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of Twenty-First Annual ACM Symposium on Theory of Computing*, pages 433–444, 1989.
18. T.G. Dietterich. "machine learning". *Annual Review of Computer Science*, 4:255–306, 1990.