# DETECTING RULE INCONSISTENCIES IN SYMBIOTIC SIMULATIONS

*Catriona Kennedy*
*Volker Sorge*
*Georgios Theodoropoulos*
School of Computer Science
University of Birmingham, UK
{C.M.Kennedy|V.Sorge|G.K.Theodoropoulos}@cs.bham.ac.uk

## ABSTRACT

*Simulation models are used to predict future or hypothetical states of an observed system. To obtain accurate simulation models one needs to ensure that the data generated by the simulation is consistent with the data obtained from the observed system. This is a central issue in the area of symbiotic simulation where there is an online feedback loop between the system and its simulation. In this paper we use association rule data mining in order to compare the two data sets. This results in rule sets for both real world and simulation data, which can then be checked for inconsistencies. We present how we formalise rules mined from different data sets in propositional logic and how we employ a SAT solving system to detect inconsistencies of different rule sets in an incremental process, which gradually incorporates rules mined during an ongoing simulation. Once an inconsistency is detected we extract the interfering simulation rules and use them to refine the simulation model.*

## 1 INTRODUCTION

In the physical sciences, symbiotic simulation (or DDDAS, *Dynamic Data Driven Application Systems*) is a method where data from a physical system is absorbed into a simulation of the system in order to continually adapt the model to the reality, if necessary making changes to the assumptions on which it is based (Darema, 2005). The simulation predictions can in turn potentially be used to determine the data to be absorbed and steer the observed system. Symbiotic simulation can be used to optimise management of real world systems in areas such as commerce, communication, or transportation by close interaction with the system being observed and simulated.

One important aspect is to ensure that the simulation models closely the real world system. As part of a project which investigates the application of symbiotic simulation to the social sciences: *AIMSS-Adaptive Intelligent Model-building for the Social Sciences* (Kennedy and Theodoropoulos (2006a), Kennedy and Theodoropoulos (2006b), Kennedy and Theodoropoulos (2005), Kennedy et al. (2007)) we have worked on a method to compare the data obtained from the real world system with the data generated by the simulation using a data mining approach. For both data sets we apply association rule mining (Agrawal and R. Srikan, 1994) to extract association rules that describe dependencies within the data. We explain this in section Sec. 2

The extracted rule sets can subsequently be checked for inconsistencies between the simulation and the real system. We use an encoding in propositional logic for the association rule set and employ a satisfiability solver to find inconsistencies. To accommodate the continuous nature of the simulation process we adopt an incremental process, which gradually incorporates rules mined during the ongoing simulation into the formalisation and consistency checking. We present the details in Sec. 3. Once an inconsistency is detected the interfering simulation rules can be extracted and used to support the refinement of the simulation model. In Sec 4, we illustrate how our approach can be applied in a simple case study from social sciences.

## 2 SYMBIOTIC SIMULATION

The accuracy of a simulation model is typically tested by verifying its predictions with respect to the available data. In other words, it should be possible for the simulation to "reproduce" the current data and act as an explanation for the observations in the real world system. The accuracy of a model can be improved as an iterative process with the following stages:

1. Formulate initial model and run simulation

2. Once the simulation has stabilised, inspect its run and determine whether it makes interesting predictions, which need to be tested
3. Collect the relevant data and analyse it
4. Determine if the simulation predictions are supported by the data
5. If the data does not support the predictions, determine whether the model should be revised. Experiment with variations of the original simulation and return to Step 2.

Traditionally steps 2-5 are done interactively, by visually inspecting the data. Within a truly DDDAS environment however, this feedback loop should ideally be automated.

To detect situations where the data refutes the predictions, we need two sets of statements:

1. General statements on the patterns and trends in the simulation.
2. General statements on the patterns and trends in the real world data.

While these two sets of statements may be assembled manually, this may be difficult, in particular in the case of the real world data, due to its complexity. For this purpose, data mining algorithms can be applied to produce either.

In order to obtain comparable sets of statements we apply the same data mining algorithm to both simulation and real world data, which in turn means that both data sets need to be similar. To generate the required datasets, we use an ontology describing what entities and events exist in the simulation and what kinds of real world data will record similar details about similar entities in the real world. A *state* of an entity is a set of values for its attributes. An *event* is any change of state of an entity and a dataset can be a sequence of events.

The definitions of entities and events are used to generate a similar dataset from the simulation. This contains a sequence of simulated events. Note that these events do not correspond to the *particular* observed system. Therefore, we do not expect to identify the individual events in the simulation within the real world data. Instead, the general patterns that are found in the simulation should also be found in the real world data, if the model is intended to explain that particular data.

To produce such generalisations, we are using *Association Rule Mining* using the Apriori algorithm, which is available in the WEKA Machine Learning package (Witten and Frak, 2005)). This algorithm is suited to large databases containing qualitative data which is often produced in social science research. Furthermore, it is "unsupervised" in the sense that predefined classes are not given. This allows the discovery of unexpected relationships.

Informally, an association rule produced by Apriori is as follows:

***if (conjunction of antecedents)*** $s_1$ ***then (conjunction of succedents)*** $s_2$ ***conf(c)***

Each antecedent and succedent has the form "attribute = value". $s_1$ and $s_2$ are known as the *support* values and $c$ is the *confidence*. The support value $s_1$ is the number of occurrences (records) in the dataset containing all the antecedents. $s_2$ is the number of occurrences of both the right and left sides together. Only those collections of items with a specified minimum support are considered as candidates for construction of association rules. The confidence is $s_2/s_1$. It is effectively the accuracy of the rule in predicting the consequences, given the antecedents. An example minimum confidence may be 0.9. The higher the support and confidence of a rule, the more it represents a regular pattern in the dataset. These measures are currently not taken into account when checking consistency. If they are relatively low, then any inconsistency would be less "strong" than it would be for rules with high confidence and high support.

## 3 DETECTING INCONSISTENCIES

In order to check the consistency of our model with the real world system we want to compare the rule sets mined for both the simulation data and the real world data, and, in case, we find inconsistencies we want to exploit this knowledge to refine the simulation model. We propose to do this using propositional logic and a SAT solver. We first describe how we formalise the sets of rules in propositional logic and then present how to employ a SAT solver to find inconsistencies between rule sets, while incrementally adding rules during the simulation.

### 3.1 Formalising Rules

The data we have both for the real world system as well as generated from the simulation is a set of records, where each record consists of a set of attributes together with their assigned values. The basic elements of our rules are therefore attribute/value pairs of the form *p*=*v*, where *p* is some attribute (or parameter) and *v* is an associated value. In each record a unique value *v* is assigned to each occurring attribute *p*. However, while *p* stays the same, *v* will vary.

The association rules we mine from the data can then be represented as

$$a_1 \wedge \ldots \wedge a_m \to c_1 \wedge \ldots \wedge c_n \qquad (1)$$

where each literal $a_i, i = 1,\ldots,m$ and $c_j, j = 1,\ldots,n$ is an attribute/value pair of the form *p*=*v*. Obviously we can simplify the overall rule set by normalising all rules to have only a single literal in the succedent. That is, a rule of the form of (1), can be replaced by *n* rules of the form $a_1 \wedge \ldots \wedge a_m \to c_j$, *j*=1,…,*n*.

We say that two literals are *related* if they have the same attribute. That is two literals *a*,*b* are related, if *a* is of the form $p_1$=$v_1$ and *b* is of the form $p_2$=$v_2$ and $p_1$=$p_2$ while $v_1$ is not necessarily equal to $v_2$. Observe that no related literals can occur in the antecedent or succedent of a rule. That is, each attribute can occur at most once in the antecedent or succedent. However, we can have related literals in the antecedent and succedent.

We call two rules *related* if they have related succedents, regardless of their antecedents. Two rules are considered *inconsistent* if they have equivalent antecedents and related succedents *p*=$v_1$ and *p*=$v_2$, but $v_1 \neq v_2$. Note that while the rules $R_1 : a_1 \wedge \ldots \wedge a_m \to p = v_1$, $R_2 : a_1 \wedge \ldots \wedge a_m \to p = v_2$ are inconsistent for $v_1 \neq v_2$, $R_1$ is nevertheless consistent, for example, with the rules $a_1 \wedge \ldots \wedge a_{m+1} \to p = v_2$ or $a_1 \wedge \ldots \wedge a_{m-1} \to p = v_2$. This expresses that if a particular condition is not met or if an additional condition is present there can be a different consequence in the model. Thus '$\to$' is not implication in the classical sense.

We therefore represent rules with a binary predicate $\digamma$ that takes a set of literals and a single literal as arguments. We then replace every single rule of the form $a_1 \wedge \ldots \wedge a_m \to c$ by a predicate $\digamma(\{a_1,\ldots,a_m\},c)$. Since we now consider the antecedents of a rule as a set of literals, thereby abolishing their order, we can easily detect and remove duplicate rules.

To postulate the consistency of a rule set we first define for each set of related succedent literals *c* a *context* as follows: Given predicates $\digamma(A_1,c_1),\ldots,\digamma(A_n,c_n)$ where each $A_i$ is a set of literals and all the $c_i$ are related with respect to an attribute *p* for *i*=1,…,*n*, i.e., every $c_i$ is of the form *p*=$v_i$, we define the *context* of the attribute *p* as $\mathbb{C}_p = \bigcup_{i=1}^{n} A_i$. Let $\mathrm{P}(\mathbb{C}_p)$ be the power set of $\mathbb{C}_p$ we then define for each $B \in \mathrm{P}(\mathbb{C}_p)$ and each $c_i, i = 1\ldots,n$ an *exclusiveness condition* of the form

$$\digamma(B,c_i) \supset \neg\digamma(B,c_1) \wedge \ldots \wedge \neg\digamma(B,c_{i-1}) \wedge \neg\digamma(B,c_{i+1}) \wedge \ldots \wedge \neg\digamma(B,c_n), \qquad (2)$$

where $\supset, \wedge, \neg$ is logical implication, conjunction, and negation, respectively.

## 3.2 Incremental Consistency Checking

In order to guarantee the consistency of the simulation model with the real world system the two association rule sets extracted from data generated by both are checked for consistency. Furthermore, since both rule sets are continuously extended by the progressing data mining as more real world and simulation data becomes available, consistency has to be checked on a regular basis.

The initial two rule sets are separately formalised as described previously, i.e., sets of rules are represented by conjunctions of predicates and exclusiveness conditions are added. We then transform both into conjunctive normal form[1] and replace all predicates by propositional variables. We can then simply employ a SAT solver (in our current implementation we employ zChaff (M. Moskewicz et al. (2001)) to check for consistency of rule sets; if the problem is satisfiable the rules are consistent, if it is unsatisfiable they are not.

Each rule set is checked for consistency separately to guarantee the correctness of the association rule mining. If both rule sets are consistent they are combined and again checked for inconsistency. Once inconsistency is detected, we search for pairs

[1] In fact, exclusiveness conditions are directly generated as pairs of negated predicates.

of inconsistent rules by first checking for each subset of rules that belong to a particular context whether they are inconsistent; this search is again conducted with zChaff. Then for all inconsistent contexts we isolate the offending rules and report them back to the simulation for model refinement.

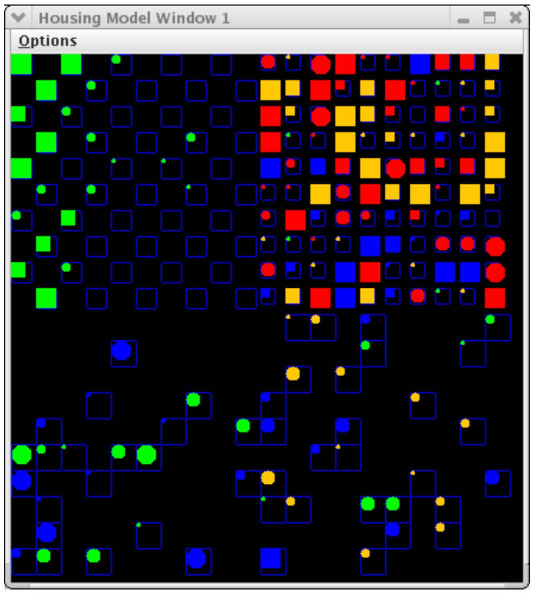
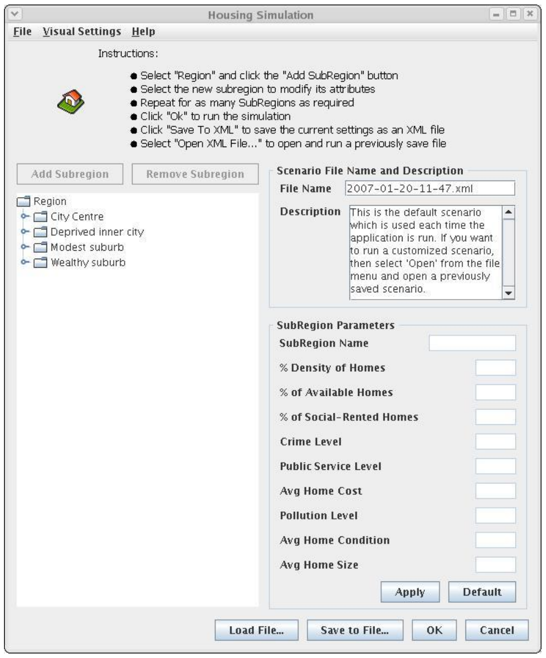


**Figure 1**: *Simulation run after n steps*



Instructions:

- Select "Region" and click the "Add SubRegion" button
- Select the new subregion to modify its attributes
- Repeat for as many SubRegions as required
- Click "Ok" to run the simulation
- Click "Save To XML" to save the current settings as an XML file
- Select "Open XML File..." to open and run a previously save file

File Name: 2007-01-20-11-47.xml

Description: This is the default scenario which is used each time the application is run. If you want to run a customized scenario, then select 'Open' from the file menu and open a previously saved scenario.

**Figure 2**: *AIMSS Configuration window*

Throughout the run of a simulation consistency of the rule sets is checked whenever new rules become available during the data mining process. New are rules are added incrementally to the problem specification and tested for consistency by formalising the new rule as additional predicate and, if necessary, adding new exclusiveness conditions. In case the new rule is of the form $\mathcal{F}(A, p = v)$ we

(i) find the context set $\mathcal{C}_p$

(ii) compute $\mathcal{C}_{p'} = A \cup \mathcal{C}_p$ and

(iii) if either $p$=$v$ is new or $\mathcal{C}_p \neq \mathcal{C}_{p'}$

we construct and add the necessary new exclusiveness conditions as given in equation (2). Similarly, if the simulation model is altered to eliminate inconsistencies, we can simply alter the problem formalisation by removing the offending rules, while still retaining the exclusiveness conditions.

## 4 A SOCIAL SCIENCE CASE STUDY

As a case study within the AIMSS project[2], we are focusing on social housing data collected by local housing authorities. This is a database of moves into the social rented sector (tenancies begun with Social Landlords) called *CORE*. Each record corresponds to a move and gives details of the previous and new situations, along with such attributes on:

(a) household details such as number of persons, total weekly income and whether they own their own home before

(b) details of the home they are moving into such as what it will cost them (weekly rent) and the type of property

(c) stated reason for move (e.g., affordability, overcrowding, health, poor condition of home, neighbour harassment) defining the state.

Events are then a sequence of moves from one home to another. For our experiments we have implemented a simplified agent-based simulation using the RePast toolki[3]. Households are represented as single agents.

The agents' behaviour is defined in the form of simple "if-then" rules that determine if and when an agent moves. At initialisation, homes are allocated randomly to regions with largest number in inner city and city centre (figure 1). Precise densities and other attributes of each region, such as crime and pollution level, public services etc., can be specified as parameters via a configuration window (as illustrated in figure 2). The simulation model is for a

[2]http://www.cs.bham.ac.uk/research/projects/aimss/
[3] http://repast.sourceforge.net/

typical housing situation on an abstract level, not a particular geographical area. Therefore, its predictions will be in the form of statements that should be true in general about households moving into social housing.

While for the CORE database some preprocessing is necessary so that it can be processed by a data mining algorithm, both datasets refer to the same kind of entities and events, and the generalised statements are therefore directly comparable and consistency-checking is feasible. An example of a typical association rule discovered in the CORE data set is:
***if incomeLevel=1, moveReason=affordability 283 then newHomeCost=1 283 c:1***

This specifies that if the income level is in the lowest bracket and the reason for moving was affordability then the rent to be paid for the new home is in the lowest bracket. The number 283 specifies how often the antecedents and succedents occur in the dataset and therefore the confidence is 1.

An example of an association rule in the simulation data set that is inconsistent with the above rule is the following

***if moveReason=affordability, incomeLevel=1 102 then newHomeCost=2 98 c:.96***

To preserve space in the formalisation we first let *p* = ***newHomeCost*** and *A*={***incomeLevel***=1, ***moveReason=affordability***}. We can then formalise the two rules as $\mathcal{F}(A, p=1)$ and $\mathcal{F}(A, p=2)$, respectively. In order to define the context of the rules we also have to take the other values of the predicate newHomeCost into account that occur in the data set, which are ***newHomeCost=3*** and ***newHomeCost=4-or-more***. Give these we can construct all the necessary exclusiveness conditions for the incremental consistency checking. In the example the relevant exclusiveness conditions to detect the two inconsistent rules are

$$\mathcal{F}(A, p=1) \supset \neg\mathcal{F}(A, p=2) \wedge \neg\mathcal{F}(A, p=3) \wedge \neg\mathcal{F}(A, p=\texttt{4-or-more})$$
$$\mathcal{F}(A, p=2) \supset \neg\mathcal{F}(A, p=1) \wedge \neg\mathcal{F}(A, p=3) \wedge \neg\mathcal{F}(A, p=\texttt{4-or-more})$$

Once zChaff detects this inconsistency the two conflicting rules can be used for the refinement of the simulation model.

## 5 CONCLUSIONS

We have presented a method for the automatic refinement of simulation models in symbiotic simulations based on association rule mining and satisfiability checking. We extract association rules for both real world and simulation data, and use a SAT solver to detect possible inconsistencies between both rule sets that can be used as feedback in the simulation model. Additional rules describing the ongoing simulation and the observed system can easily be integrated in an incremental process as soon as they become available.

In addition to finding inconsistencies, the current implementation also enables simplification of the rule set by removing redundancies as well as by grouping rules with respect to related succedents. This aids the presentation of the rules and thus their interpretation by the user.

While in the current state of our system the satisfiability problems are still fairly simple, we have already had promising results finding inconsistencies for the simulation in our housing case study. We are currently experimenting with ways to incorporate the knowledge into the simulation model. Furthermore, our work is a first step to further applications of SAT solving in the context such as finding minimal rule sets or deriving consequences from mined association rules for better prediction of system behaviour.

## REFERENCES

M. Moskewicz and C. Madigan and Y. Zhao and L. Zhang and S. Malik, (2001), Chaff: Engineering an efficient SAT Solver, Proceedings o fthe DAC-2001 Conference, pp530 – 535.

I. Witten and E. Frak (2005), Data Mining: Practical Machine Learning Tools and Techniques, Elsevier, 2005

R. Agrawal and R. Srikan (1994), Fast Algorithms for Mining Association Rules in Large, Proceedings of the International Conference on Very Large Databases, pp. 478–499

C. Kennedy and G. Theodoropoulos, (2006a) Intelligent Management of Data Driven Simulations to Support Model Building in the Social Sciences, Proceedings of ICCS 2006, LNCS 3993, pp. 562–569, 2006.

C. Kennedy and G. Theodoropoulos, (2006b) Adaptive Intelligent Modelling for the Social Sciences: Towards a Software Architecture", Technical Report CSR-06-11, University of

Birmingham, School of Computer Science, October 2006.

C. Kennedy and G. Theodoropoulos, Towards Intelligent Data-Driven Simulation for Policy Decision Support in the Social Sciences, Technical Report CSR-05-9, University of Birmingham, School of Computer Science, October 2005

P. Lee, Ed Ferrari, C. Kennedy, G. Theodoropoulos and C. Skelcher, Assisted Model Building in the Social Sciences using Data Driven Simulation, Proceedings of 2nd International Conference in e-Social Science, Manchester, UK. June 2006.

C. Kennedy, G. Theodoropoulos, E. Ferrari, P. Lee, and C. Skelcher, Towards an Automated Approach to Dynamic Interpretation of Simulations, Asia Modelling Symposium 2007, University in conjunction with Thailand’s 11th Annual National Symposium on Computational Science and Engineering (ANSCSE-11), Prince of Songkla University, Phuket Campus, 27 - 30 March 2007.

F. Darema, Grid Computing and Beyond: The Context of Dynamic Data Driven Applications Systems, (2005) Proceedings of the IEEE: Special Issue on Grid Computing, Vol. 93, No. 3, pp.692–697, 2005.