


Model Revision of Logical Regulatory Networks Using Logic-Based Tools

Filipe Gouveia¹

INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
Rua Alves Redol 9, 1000-029, Lisboa, Portugal
filipe.gouveia@tecnico.ulisboa.pt
 <https://orcid.org/0000-0003-1852-2782>

Inês Lynce²

INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
Rua Alves Redol 9, 1000-029, Lisboa, Portugal
ines.lynce@tecnico.ulisboa.pt
 <https://orcid.org/0000-0003-4868-415X>

Pedro T. Monteiro³

INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
Rua Alves Redol 9, 1000-029, Lisboa, Portugal
pedro.tiago.monteiro@tecnico.ulisboa.pt
 <https://orcid.org/0000-0002-7934-5495>

Abstract

Recently, biological data has been increasingly produced calling for the existence of computational models able to organize and computationally reproduce existing observations. In particular, biological regulatory networks have been modeled relying on the Sign Consistency Model or the logical formalism. However, their construction still completely relies on a domain expert to choose the best functions for every network component. Due to the number of possible functions for k arguments, this is typically a process prone to error. Here, we propose to assist the modeler using logic-based tools to verify the model, identifying crucial network components responsible for model inconsistency. We intend to obtain a model building procedure capable of providing the modeler with repaired models satisfying a set of pre-defined criteria, therefore minimizing possible modeling errors.

2012 ACM Subject Classification Computing methodologies → Logic programming and answer set programming

Keywords and phrases Logical Regulatory Networks, Model Revision, Answer Set Programming, Boolean Satisfiability, Logic-based tools

Digital Object Identifier 10.4230/OASICS.ICLP.2018.23

1 Introduction

Modeling biological regulatory networks is particularly useful to test hypotheses and to identify predictions *in silico*. With this aim, different qualitative formalisms have been introduced to model, analyze and simulate regulatory networks and their behaviors. However,

¹ Fundação para a Ciência e a Tecnologia (FCT) PhD grant SFRH/BD/130253/2017.

² National funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

³ Fundação para a Ciência e a Tecnologia (FCT) project grant PTDC/EEI-CTP/2914/2014.



© Filipe Gouveia, Inês Lynce, and Pedro T. Monteiro;
licensed under Creative Commons License CC-BY

Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018).

Editors: Alessandro Dal Palu', Paul Tarau, Neda Saeedloei, and Paul Fodor; Article No. 23; pp. 23:1–23:10

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the simulation and analysis of such behaviors is hindered by the combinatorial explosion of the qualitative state space. To tackle this problem, formal verification techniques have been introduced in Systems Biology. These techniques include model-checking techniques to automatically verify reachability properties [14], model reduction techniques to reduce the size of the generated dynamics [15], SAT-based approaches to identify attractors [4], among others [17].

Given a complete model of a regulatory network, newly acquired experimental data may render it inconsistent, forcing the model to be revised and updated. The process of review and update a model is called model revision, which is still mainly a manual task performed by a modeler, typically an expert in the domain, and therefore prone to error.

Approaches to model revision relying on the Sign Consistency Model (SCM) have been implemented using logic-based tools such as Answer Set Programming (ASP)[8] and Boolean Satisfiability (SAT)[10]. However, the SCM lacks in expressiveness for regulatory functions, as it is based in sign algebra. This work aims to extend current approaches for model revision to the Logical formalism, and to provide a semi-automatic tool to assist the modeler throughout the model definition process [21].

An overview of some of the key concepts of regulatory networks is given in Section 2. In Section 3 it is mentioned some of the work done in System Biology, regarding regulatory networks. Section 4 describes the logic-based approach for Model Revision. Section 5 concludes the document with an overview of the directions of the future work.

2 Regulatory Networks

A biological regulatory network is a set of proteins and genes, that interact with each other or with other substances in the cell. Qualitative models have proven to be well adapted for the modeling of systems where quantitative information is generally incomplete or noisy. Typically, network components only affect other components above some concentration level. In this way, it is possible to consider discrete variables to model regulatory networks, corresponding to different levels of concentration, e.g. active/inactive.

2.1 Logical Model

Logical models were used to represent regulatory networks by Kauffman in 1969 [12], and Thomas in 1973 [20].

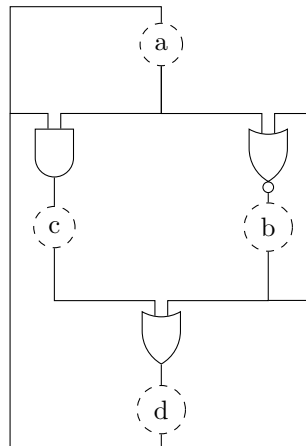
In the Logical Model the components of the network are represented by Boolean variables. A Boolean variable can either be *True* (1, on, active) or *False* (0, off, inactive). If a component in a regulatory network is represented by a Boolean variable, then it has value *True* if it is present (or activated), and it has value *False* if it is absent (or inhibited).

Moreover, the interactions between components are described as Boolean functions [20]. This will allow to determine the state of a component based on the presence or absence of other components.

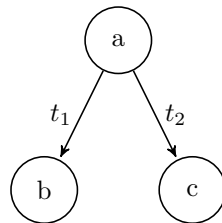
A Logical Model can be represented with a logical circuit since nodes have a Boolean value and regulatory functions are Boolean functions, as shown in Figure 1.

Figure 1 illustrates an example of a logical model with the correspondent regulatory functions. With this representation we can verify that, for example, component *c* is regulated by components *d* and *a*, and its regulatory function is a logical AND from these two inputs.

The (Boolean) Logical Model can be generalized [21]. It is possible to consider more than two values for each variable. For example, considering Figure 2, we can have a variable *a* that affects *b* above a concentration level threshold t_1 , but only affects *c* above a concentration level threshold $t_2 > t_1$. In this case variable *a* can have three possible values:



■ **Figure 1** Example of a Logical Model represented as a logical circuit.



■ **Figure 2** Example of a Generalization of the Logical Model.

- 0: concentration level below t_1 (not affecting any other variable);
- 1: concentration level between t_1 and t_2 (only affecting variable b);
- 2: concentration level above t_2 (affecting variables b and c);

Formally, we can define a Logical Model as a tuple (G, K) where:

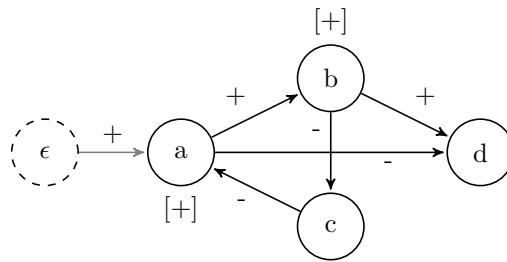
- $G = \{g_1, g_2, \dots, g_n\}$ is the set of components of the network. Each g_i is associated with an integer value in $\{0, \dots, max_i\}$, representing the concentration level of the component. The state of the network is thus defined as a vector $s \in S = \prod_{g_i \in G} \{0, \dots, max_i\}$.
- $K = \{K_1, K_2, \dots, K_n\}$ is the set of regulatory functions where K_i is the regulatory function of g_i and $K_i : S \rightarrow \{0, \dots, max_i\}$.

If all $max_i = 1$, then we have a Boolean Logical Model, since each $g_i \in \{0, 1\}$.

2.2 Probabilistic Boolean Networks

In a logical model, each component regulated by k other components can have 2^{2^k} possible regulatory Boolean functions. Additionally, in some cases experimental data is insufficient or there is incomplete knowledge to choose a single regulatory function, where several candidates are possible. In other words, possibly several regulatory functions could explain the experimental data. With this in mind, the logical model was extended in order to account for the uncertainty of the regulatory functions [18].

In a Probabilistic Boolean Network (PBN), each component has several regulatory functions, each with a given probability associated. These probabilities are determined based on the data available, such that it is compatible with prior knowledge of the network. Then, at each time step, and for each component, a regulatory function is selected according to the correspondent probabilities, in order to determine its target value.



■ **Figure 3** Example of a Sign Consistency Model. Observed components a and b are labeled with the correspondent observation. Component ϵ represents an external stimulus.

2.3 Sign Consistency Model

Siegel *et al.* proposed a Sign Consistency Model (SCM) [19]. In this approach, it is only considered the difference in the expression levels between two situations: a value increase, or decrease.

The SCM is usually represented by a graph where each node represents a biological component, with a value $+$ (increase of concentration) or $-$ (decrease of concentration). The edges in the graph represent interactions between components and can be labeled “ $+$ ” or “ $-$ ”. An edge with label “ $+$ ” (“ $-$ ”) from a to b means that an increase of the concentration of a increases (decreases) the concentration of b .

Also, a component can be considered an *input*, having a stimulation from the exterior world (outside the regulatory network). If a node is an *input* then its regulatory function can be ignored, as there is an exterior stimulation increasing its concentration. In some representations, an extra generic node ϵ is added to represent the exterior world. For each *input* node, an edge is added from ϵ to that node.

The regulatory functions are then based on the sign algebra, where the value of each component is the sum of the products between the value of each regulator and the corresponding edge.

Figure 3 illustrates an example of a Sign Consistency Model of a network, where node a is an input, and therefore have an input edge from the generic node ϵ that represents the exterior world. Nodes a and b are observed nodes, where an increase of concentration was observed. In this example, node c is expected to have a negative ($-$) sign because it only has one regulator b , which has a negative interaction with c ($c = b \times (b \rightarrow c) = (+) \times (-) = -$). However in this example, node d receives a positive and a negative interaction. in this case we say that we have a competition and d can assume either value.

3 Related Work

The analysis and verification of biological regulatory networks provide opportunities for the application of several methodologies. From network identification and parametrization, model verification, attractors determination or to model revision. In this section, some of the main methods from the last decade are described, as well as the corresponding problem and technology.

3.1 Network and Model Inference

Building computational models to correctly represent regulatory networks is of great importance. In order to build such model, one first needs to infer the network topology from a given set of experimental data. Some of the difficulties of this task relies on the few samples

of observational data and in the incompleteness and inaccuracy of the experimental data. Then, one also needs to infer, for each component, the associated regulatory functions (model inference).

In regulatory networks inference, several statistical learning techniques are commonly used [2, 7]. Also, logic-based tools have been successfully used to learn biological models. *Caspo* [11] is a tool to identify the complete family of feasible models from a training Boolean logical model from prior knowledge and experimental data.

3.2 Reachability Verification

Given a model and a set of experimental data, it is interesting to verify if the model can explain the results obtained in the experiment. In particular, one may verify if the model is capable of generating behaviors from a set of initial states to a set of target states. These behaviors are typically represented by a State Transition Graph (STG), where nodes represent states of the network, and edges represent possible transitions between states. The generation of this STG can be made synchronously or asynchronously. In the synchronous approach, in a given state of the STG, all components can update their value simultaneously, i.e., each state as a single successor. In the asynchronous approach, in a given state of the STG, only one component can update their value to a successor state, i.e., each state has as many successors as components changing their values.

Model checking consists in the verification if a model satisfies a given (set of) property [3], and has been successfully used for the verification of regulatory networks. Here, biological observations are encoded in temporal logic formulas, and a model checker is used to verify the existence of particular behaviors [14].

Also of interest, is to know how can a system be influenced in order to avoid reaching unsafe or undesired states. Recently, the work in [6] introduces the notion of *bifurcation*, transitions after which a given goal is no longer reachable. This work presents a method using Answer Set Programming, to identify bifurcations given a model represented as a discrete finite-state of interacting components. However, since this method relies on under/over approximations, is not complete, i.e., does not guarantee the identification of all the bifurcations.

3.3 Attractors Identification

A key property of the dynamics of a regulatory network are *attractors*, which typically denote subsets of states of biological interest. There are two types of attractors: point attractors and cycle attractors. A point attractor, or a stable state, is a state from which there is no transition to any other state in the STG. A cycle attractor is a set of states, whose sequence repeats over time, from which no transition can leave, i.e., a terminal strongly connected component in the STG.

An efficient approach to determine point attractors in (multivalued) logical models uses Multi-values Decision Diagrams (MDDs) [16]. Also, some approaches consider the identification of point and cycle attractors in synchronous dynamics. The work in [5] uses Answer Set Programming (ASP) and allows the determination of all attractors considering a Markovian program in order to overcome the challenge of determining the number of time-steps needed to achieve an attractor. The work in [4] uses a SAT based bounded model checker to determine all the attractors of the network by incrementally determining the attractors of a given length.

3.4 Reduction

It is often the case where the generation of the network dynamics is intractable for large and complex regulatory networks, due to the state space combinatorial explosion. Reduction techniques can then be applied in order to reduce the model, and therefore the generated state space. It has been shown that reduction methods can be successfully applied preserving some dynamical properties of the network, such as attractors [15].

4 Model Revision Approach

During the iterative model construction procedure, as new data is acquired, the current model may not be able to explain the new data, and therefore need to be revised. Revision processes capable of suggesting addition/removal of networks interactions, and changes to variable values in order to make a model consistent with the available data have been proposed [13]. An approach was proposed considering the SCM and developed using the Answer Set Programming (ASP) paradigm [8]. Also, an approach was proposed using MaxSAT, a SAT extension used to solve optimization problems [10]. However, the SCM formalism relies on a simple rule regarding regulatory functions.

The logical formalism [20] has been widely used to model biological networks, and have been successfully implemented using ASP [9] and SAT [1], allowing to model the regulatory functions with increased expressiveness w.r.t. the SCM. Model repair usually operates under a minimal assumption as there can be several ways to make a model consistent. Such optimization criteria can be regarding the number of atomic repair operations [8, 10] or considering some properties found in the literature [13]. Nevertheless, existing approaches typically rely on repair operations that potentially change the topology of the network, invalidating previous domain knowledge.

As mentioned in Section 2, there can be several regulatory functions that can explain the experimental data. Avoiding changing the topology of the network and change regulatory functions leads to a minimal impact on the truth table of the variables of the model, and therefore a smaller impact on the associated dynamics.

Our idea is to develop a model revision procedure capable of building a consistent model iteratively as new data is acquired, relying on the logical formalism. Moreover, it is desired to avoid changing the topology of the network, and try to explain possible causes of inconsistencies with regulatory functions.

On a first phase of the work, one should be able to verify the consistency of a given model with a set of experimental data, i.e., if the model can explain the experimental data obtained. Model checking techniques should be used for this purpose. It is intended to implement this using different logic based tools, such as ASP, SAT and MaxSAT, in order to make a comparison with respect to the easiness of representation and computational efficiency.

On a second phase, if a model is not consistent with the experimental data, the causes of such inconsistencies must be identified. This is closely related to the identification of Minimal Unsatisfiable Subsets (MUSes) in SAT formulas, and therefore SAT-based tools should be used in the process. As there can be multiple concurrent reasons to explain the existence of inconsistencies, a biological meaningful measure should be provided in order to rank the possible explanations to be presented to a modeler.

On a final phase, considering the most plausible cause for inconsistency, a procedure for model revision should be defined. For this, SAT-based tools for the identification of Minimal Correction Subsets (MCSes) should be considered. This model revision process should be iterative, considering that multiple reasons for inconsistency may exist. We will first try

■ **Listing 1** Example of input.

```

edge(c1,c2,0).
edge(c1,c3,1).
edge(c2,c1,1).
edge(c2,c3,1).
edge(c4,c2,1).
edge(c4,c3,0).

functionOr(c1,1).
functionAnd(c1,1,c2).

functionOr(c2,1).
functionAnd(c2,1,c1).
functionAnd(c2,1,c4).

obs_vlabel(c1,1).
obs_vlabel(c2,0).
obs_vlabel(c3,0).
obs_vlabel(c4,0).

functionOr(c3,1..2).
functionAnd(c3,1,c1).
functionAnd(c3,2,c2).
functionAnd(c3,2,c4).

```

to explain the causes of inconsistencies with regulatory functions. However, changing the regulatory functions may not be sufficient, and therefore one may need to consider changing the topology of the network. To achieve this, an iterative approach will be considered where different causes of inconsistency are taken into account.

In the revision process, not only the model consistency must be taken into account, but also other known properties about the network must hold, such as the existence of known attractors and its reachability. As the number of possible states for a network increases exponentially with the number of components, guaranteeing the existence of the known attractors, for example, can be a difficult task. For this, model reduction techniques may be necessary.

We start by considering only monotone non-degenerate functions. In a monotone function, each regulator has only one role, i.e., it is either strictly positive or negative. In a non-degenerate function, all regulators are functional, i.e., all regulators have an influence in the regulatory function.

Currently, we have an Answer Set Programming approach implemented for the logical formalism, and we are able to verify the consistency of a model given some experimental data at steady state, i.e., without considering any dynamics. Moreover, we are able, in case of inconsistency, to identify the regulatory functions that can explain such inconsistencies. We are working on the process of repairing such functions in order to validate if the proposed model solutions become consistent.

We represent the logical model as a directed graph and the regulatory functions in disjunctive normal form (DNF). As we only consider monotone functions and, therefore, each regulator only has one role (positive interaction or negative interaction), this role is defined by the edge. A positive (negative) edge represents a positive (negative) interaction. An example is presented in Listing 1 with the representation of the model and the observations.

The predicate `edge(A,B,S)` represents an edge from A to B with sign S. Predicate `functionOr(A,C)` indicates the number of clauses (C) in the regulatory function of A in the DNF. Predicate `functionAnd(A,C,B)` indicates that the clause C of the regulatory function of A contains variable B. The observations are represented by the predicate `obs_vlabel(A,S)`, which means that value S was observed in node A.

■ **Listing 2** Consistency check in Answer Set Programming.

```

sign(0;1).                                complement(T,S):-sign(S),sign(T),T!=S.

vertex(V):-edge(V,_,_).                    vertex(V):-edge(_,V,_).

% generate
1{vlabel(V,S):sign(S)}1:-vertex(V).
{r_gen(V)} :- vertex(V).                    {r_part(V)} :- vertex(V).

:-vlabel(V,S), obs_vlabel(V,T),complement(S,T).

% functions
% one positive or negative contribution in a clause
onePositive(V,Id):-functionAnd(V,Id,V2),edge(V2,V,S),vlabel(V2,S).
oneNegative(V,Id):-functionAnd(V,Id,V2),edge(V2,V,S),vlabel(V2,T),
                    complement(S,T).

% none negative contribution in a clause
noneNegative(V,Id):-onePositive(V,Id),not oneNegative(V,Id).

vlabel(V,1):-1{noneNegative(V,Id):functionOr(V,Id)},vertex(V),
              not r_part(V).
vlabel(V,0):-{noneNegative(V,Id):functionOr(V,Id)}0,vertex(V),
              not r_gen(V).

repair(f,V) :- r_gen(V).                    repair(f,V) :- r_part(V).
#minimize {1,V : repair(_,V)}.

```

The main idea behind the encoding presented in Listing 2 is that each node of the network (*vertex*) must have exactly one label that represents the expected value (*vlabel*), and it is not possible to have a label different from the observation. Each label is determined based on the contributions of each regulator in the associated regulatory function. To allow determining possible causes of inconsistencies, we defined the predicates *r_gen* and *r_part* indicating that a regulatory function should be generalized or particularized, respectively, justifying the inconsistency of the model. In order to achieve this, we allow a label of a vertex to be different than expected given the regulatory function, if that function is a possible cause of inconsistency.

5 Conclusions and Future Work

Qualitative formalisms have been used whenever information is scarce. In particular, the logical formalism has proved successful to model complex biological networks. Nevertheless, the construction of such models is still mainly a manual task, and therefore prone to errors and to interpretations of a specific modeler. Here, we focus on the problem of model revision, i.e., to assist the modeler in the process of revising the model associated functions in order to render the model consistent with the existing and new data.

Here, we propose to consider the logical formalism limiting to the set of monotone non-degenerate functions. Also, we start by verifying the consistency of models at steady state, i.e., without considering any dynamics. We consider an Answer Set Programming approach to identify which nodes are the causes for model inconsistency.

We intend to follow the work plan described in the previous section, and be able to present a procedure and corresponding tool capable of building a consistent model iteratively as new data is acquired.

References

- 1 Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.
- 2 Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Biocomputing 2000*, pages 418–429. World Scientific, 1999.
- 3 Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.
- 4 Elena Dubrova and Maxim Teslenko. A SAT-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(5):1393–1399, 2011.
- 5 Timur Fayruzov, Jeroen Janssen, Dirk Vermeir, Chris Cornelis, and Martine De Cock. Modelling gene and protein regulatory networks with answer set programming. *International journal of data mining and bioinformatics*, 5(2):209–229, 2011.
- 6 Louis Fippo Fitime, Olivier Roux, Carito Guziolowski, and Loïc Paulevé. Identification of bifurcation transitions in biological regulatory networks using Answer-Set Programming. *Algorithms for Molecular Biology*, 12(1):19, 2017.
- 7 Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805, 2004.
- 8 Martin Gebser, Carito Guziolowski, Mihail Ivanchev, Torsten Schaub, Anne Siegel, Sven Thiele, and Philippe Veber. Repair and Prediction (under Inconsistency) in Large Biological Networks with Answer Set Programming. In *KR*, 2010.
- 9 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(3):1–238, 2012.
- 10 João Guerra and Inês Lynce. Reasoning over biological networks using maximum satisfiability. In *Principles and Practice of Constraint Programming*, pages 941–956. Springer, 2012.
- 11 Carito Guziolowski, Santiago Videla, Federica Eduati, Sven Thiele, Thomas Cokelaer, Anne Siegel, and Julio Saez-Rodriguez. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. *Bioinformatics*, page btt393, 2013.
- 12 Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177, 1969.
- 13 Elie Merhej, Steven Schockaert, and Martine De Cock. Repairing inconsistent answer set programs using rules of thumb: A gene regulatory networks case study. *International Journal of Approximate Reasoning*, 83:243–264, 2017.
- 14 Pedro T Monteiro, Wassim Abou-Jaoudé, Denis Thieffry, and Claudine Chaouiya. Model Checking Logical Regulatory Networks. *IFAC Proceedings Volumes*, 47(2):170–175, 2014.
- 15 Aurélien Naldi, Elisabeth Remy, Denis Thieffry, and Claudine Chaouiya. Dynamically consistent reduction of logical regulatory graphs. *Theoretical Computer Science*, 412(21):2207–2218, 2011.
- 16 Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. In *CMSB*, volume 7, pages 233–247. Springer, 2007.
- 17 Loïc Paulevé. Reduction of Qualitative Models of Biological Networks for Transient Dynamics Analysis. *IEEE/ACM transactions on computational biology and bioinformatics*, 2017.

23:10 Logic-Based Approach for Model Revision

- 18 Ilya Shmulevich, Edward R Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.
- 19 Anne Siegel, Ovidiu Radulescu, Michel Le Borgne, Philippe Veber, Julien Ouy, and Sandrine Lagarrigue. Qualitative analysis of the relation between DNA microarray data and behavioral models of regulation networks. *Biosystems*, 84(2):153–174, 2006.
- 20 René Thomas. Boolean formalization of genetic control circuits. *Journal of theoretical biology*, 42(3):563–585, 1973.
- 21 René Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of theoretical biology*, 153(1):1–23, 1991.