PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 39, NO. 1, PP. 39–52 (1996)

CONSTRAINT-BASED DIAGNOSIS ALGORITHMS FOR MULTIPROCESSORS

András PETRI*, Péter URBÁN*, Jörn ALTMANN** Mario DAL CIN**, Endre SELÉNYI*, Károly TILLY* and András PATARICZA*

* Department of Measurement and Instrumentation Engineering
Technical University of Budapest
Műegyetem rkp. 9.
H–1521 Budapest, Hungary
email: pataric@mmt.bme.hu

** Department of Computer Science III
University of Erlangen-Nürnberg
Martensstr. 3
91058 Erlangen, Germany
email:jnaltman@informatik.uni-erlangen.de

Received: May 23, 1995

Abstract

In the latest years, new ideas appeared in system level diagnosis of multiprocessor systems. In contrary to the traditional diagnosis models (like PMC, BGM, etc.) which use strictly graph-oriented methods to determine the faulty components in a system, these new theories prefer AI-based algorithms, especially CSP methods. Syndrome decoding, the basic problem of self-diagnosis, can easily be transformed into constraints between the state of the tester and the tested components. Therefore, the diagnosis algorithm can be derived from a special constraint solving algorithm. The 'benign' nature of the constraints (all their variables, representing the fault states of the components, have a very limited domain; the constraints are simple and similar to each other) reduces the algorithm's complexity so it can be converted to a powerful distributed diagnosis method with a minimal overhead. Experimental algorithms (using both centralized and distributed approach) were implemented for a Parsytec GC massively parallel multiprocessor system.

Keywords: self-diagnosis, multiprocessor systems, constraint satisfaction.

1. Introduction

1.1 Traditional Methodology of Self-Diagnosis

The construction of dependable systems is hardly possible without the application of some forms of self-checking. Therefore different models and algorithms were developed for system level (self-)diagnosis. The majority

of them are based on graph theory derived from the first so-called 'system level models' published in the mid-sixties.

These introductory models (PMC for symmetric and BGM for asymmetric test invalidation) are the well-known and most widely used ones. Their mathematical apparatus is simple and well-elaborated; both theoretical investigations and practical implementations proved their usefulness. However, these models have some implicit limitations preventing their use in many important fields of application. The test invalidation model is oversimplified in order to assure a proper mathematical treatment decreasing the level of reality of the models and reducing their usability.

The rapid development of electronic technology and computer architectures modified radically the basic assumptions used originally in the diagnostic model:

- the fault rates are much lower and majority of the faults is transient;
- the complexity of other system components is comparable with the complexity of CPUs;
- the complexity of systems and the number of the computing elements have been drastically increased.

Most insufficiencies result from the assumption of a homogeneous system; i.e. all system components have identical test invalidation properties. This assumption reduces effectively the range of applicability due to the growing practical importance of inhomogeneous multiprocessor systems.

1.2 Required Features of a New Self-Diagnosis Method

The new requirements, resulting from the latest results in multiprocessor system design, characterize the expected features of a proper, general purpose self-diagnosis method:

- it should be applicable in homogeneous systems as well as in inhomogeneous ones (different components with different test invalidation models are to be considered);
- the diagnostic resolution should only loosely depend on the actual system topology and/or test invalidation model (the currently used methods have serious restrictions on the system topology due to the use of rigid, inadaptive algorithms);
- the algorithm should extract all the useful information from the elementary diagnostic results (e.g. for estimating the level of diagnosis at run-time);
- it should cope with the latest massively parallel processor systems with several hundreds or even thousands of system components, thus

the algorithm should have an excellent efficiency even for a very high number of units under test.

These requirements need a new approach. A generalized test invalidation model for syndrome decoding and diagnosis in inhomogeneous systems is published in [7]. This model contains all sufficient and necessary conditions of one-step and sequential diagnosis for the different test invalidation models. However, its mathematical apparatus is suboptimal (it uses complex matrix operations, e.g. computation of transitive closure); therefore the efficiency of the algorithm becomes a crucial factor in large-scale system diagnosis.

The most important step of self-diagnosis is basically the process of finding the possible fault states of system components based on the syndrome information. A systematic search method is required for effective syndrome decoding. Many applications even demand on-the-fly diagnosis for maximal performance; it requires a diagnosis method that is able to identify the fault states of some units from partial syndrome information. This is hardly achievable with traditional algorithms.

1.3 The Use of AI-based Methods and Algorithms

The main intention of 'artificial intelligence' (AI) methods is to find efficient solutions for difficultly solvable (to be more precise, generally NP-complete) or hard to represent problems. This gives a way to handle many practical but earlier unmanageable problems.

Many well-elaborated, efficient and widely tested AI-based algorithms have been developed over the last years. A group of them, the CS (Constraint Satisfaction) methods seem especially useful for a special self-diagnosis model [5].

Constraint satisfaction problems can be described as a set of variables and a set of relations between them. The solution of a CS problem is a particular set (or all the possible sets) of values given to the variables which satisfy all the relations.

The application of CS methods has already proven very attractive on fields closely related to system level diagnosis. For example, CS-based automated test pattern generation is presented in [8].

1.4 Formulating a Self-Diagnosis Problem as a CSP

There are many similarities between methods of self-diagnosis and constraint satisfaction. Actually the final goal is very similar: we want to know the fault state of the system components that conforms to our diagnos-

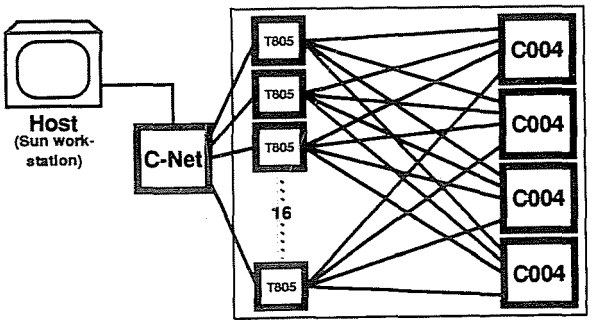

Fig. 1. The structural layout of the Parsytec GCel (1 cluster)

tic model, the test invalidation rules and the actual test results (syndrome pieces). These restrictions can be represented by binary relations between the state of processors in a test pair. The exact relation is determined by the test result, thus a set of relations can be built from the syndrome information and can be applied to find the possible fault states of the system.

The use of relations instead of logical functions is advantageous, because the diagnostic uncertainty appearing in some test invalidation (e.g. in the PMC model a faulty unit may be tested as good by another faulty unit) can be handled as well. The relations can be handled by a uniform mechanism, independently from the invalidation rules, system topology, the considered number of faults and special properties of syndromes. So this representation is very flexible and is applicable on a wide range of systems.

Therefore a self-diagnosis problem can be reformulated very easily to a constraint satisfaction problem. The variables of the CSP represent the fault state of the system components. The constraints represent the restrictions from the model by the test invalidation relations and by the actual syndrome part. If one-pass diagnosis is allowable, a static binary CSP is produced. In the case of diagnosis on the fly (performing a preliminary diagnostic process during the collection of syndrome parts), only a few syndrome pieces are present so the complete set of relations cannot be built at the beginning. Every incoming test result, however, reduces the solution space of possible fault states. The previously constructed relations (constraints) remain valid, just new constraints have to be added. Therefore a kind of dynamic CSP can represent this case.

This reformulation gives a way to handle self-diagnosis problems very comfortably, with the well-elaborated toolset of CSP solution methods. With a sufficient diagnostic model, a very flexible method can be constructed.

2. Implementation Environment

The experimental implementation of the CSP-based diagnosis algorithm was created on a Parsytec GCel massively parallel reliable multiprocessor machine (*Fig. 1*). The computing elements are Inmos T805 transputers. They are grouped by 16 to clusters; these clusters are the basic building blocks of a machine that is scalable up to 16384 transputers.

Each transputer has 4 physical data links. These are connected to C004 routing chips that provide a very fast, reliable and deadlock-free message routing and connection management. Each cluster has 4 routing chips; every C004 chip has 32 connection ports (17 for the internal interconnection of the cluster, 8 for the connection between clusters and 7 for I/O control and other purposes).

Despite of the 4 physical data connections, each transputer can communicate with an arbitrary number of other transputers via so-called virtual links. These are managed by a special unit of the T805 by multiplexing data packets on the physical connections. The configurable crossbar switches make allocation of virtual links very easy; complete virtual topologies are supplied with development libraries. The physical topology of each cluster is a 4×4 two-dimensional mesh.

Peripheral I/O management is done by a separate host machine (e.g. a Sun workstation) connected to the Parsytec GCel machine.

The machine has a Control Network (C-Net): every crossbar switch has a direct link to a special group of transputers directly connected to the host machine. This separate group is used for dynamic configuration management and job control.

Table 1
The possible fault states in the fault model

Unit	Fault state and its notation		Behaviour
Processor	fault free	0_p	correct operation
	faulty (incorrect computation fault)	1_p	incorrect test result evaluation
	dead (crash fault)	c_p	no communication
Data link	alive	$L_{p,R}$	correct message transfer
	broken	$\overline{L}_{p,R}$	no message transfer
Router	fault-free	–	correct operation
	single port fault	$L_{p,R}$	no message transfer via the faulty port
	dead	m_R	all ports are faulty

2.1 The Centralized Approach

2.1.1 The Fault Model

In order to validate the concepts described above a simple fault model was developed for the Parsytec machine and a syndrome decoding algorithm was implemented using the standard test procedures available. The fault model used includes the faults in interprocessor links and crossbar switches as well, additionally to the processor faults.

Testing of these system components is done by mutual time-out protected periodical 'I'm alive' messages between neighbouring processors. The asynchronous communication mode is used for message exchange because it does not block the sender processor (time-out detection is possible).

The considered fault states for the components are enumerated in *Table 1.* The possible test results are: good (the 'I'm alive' message was correctly received within the time-out limit), faulty (the 'I'm alive' message was received within the time-out limit but was corrupted) or dead (no message was received).

The diagnostic kernel of the algorithm is running on the host in a centralized form. The processors have a separate node-host data connection to the host machine (via the C-Net) independent from the routing chips, and thus can be considered fault-free.

The developed algorithm is for intra-cluster diagnosis (we assume only a single routing chip between 2 processors) but it can be easily extended hierarchically to diagnose the whole Parsytec machine.

2.1.2 The Test Invalidation Scheme and Implication Rules

The PMC type (symmetric) test invalidation was used for the algorithm. It was mandatory due to the testing with 'I'm alive' messages; other, more sophisticated test methods make more optimistic invalidation possible.

Syndrome decoding is driven by implication rules, represented by constraints. They originate from the system structure (fault domination rules), the test invalidation model and the actual syndrome pieces (*Fig. 2*).

All the constraints are binary to achieve greater simplicity: the test results (syndrome bits) are eliminated from them as variables, only the fault state of the tester and the tested component are variables. However, test results are already known before the syndrome decoding starts, thus

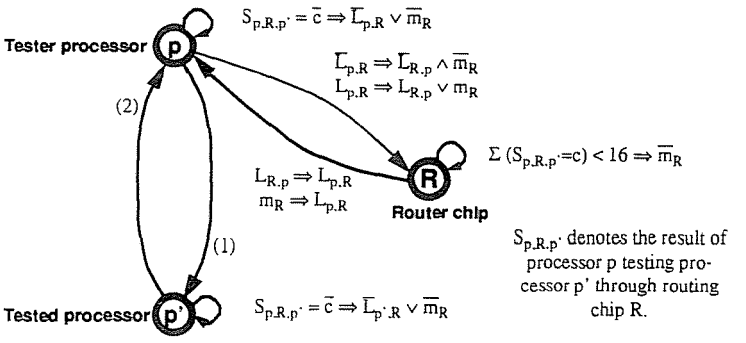

Fig. 2. Constraints resulting from the system structure and from the test invalidation model

the constraint network can be updated incorporating the newly received syndrome part as a constant constraint.

The constraints from the implication rules are as follows:

- (1) Forward (implication from the state of the tester to the state of the tested)
  - $S_{p,R,p'} = 0 \wedge 0_p \Rightarrow 0_{p'}$ (if the tester processor is fault-free and the test result is 'good' then the tested processor is also fault-free);
  - $S_{p,R,p'} = 1 \wedge 0_p \Rightarrow 0_{p'}$:
  - $S_{p,R,p'} = c \wedge 0_p \Rightarrow L_{p,R} \vee m_R \vee L_{p',R} \vee c_{p'}$.
- (2) Backward (from the state of the tested to the state of the tester)
  - $S_{p,R,p'} = 0 \wedge 1_{p'} \Rightarrow 1_p$ (if a faulty unit is tested as good then the tester is faulty);
  - $S_{p,R,p'} = 1 \wedge 0_{p'} \Rightarrow 1_p$:
  - $S_{p,R,p'} = c \wedge \bar{c}_{p'} \Rightarrow L_{p,R} \vee m_R \vee L_{p',R} \vee \bar{0}_{p'}$.

2.1.3 Implementation of the Algorithm

The self-diagnosis algorithm is implemented in two parts: the low-level tester mechanism (sending and receiving the 'I'm alive' messages) runs on the Parsytec machine, while the diagnosis kernel runs on the host machine.

The test process is controlled by the host: it initiates the test sequence (starts the Parsytec processors to exchange 'I'm alive' messages), collects the results from the processors, maintains the dynamic CSP data structure, runs the CSP solver algorithm and displays the results. The two program parts communicate through sockets.

The CSP solver. The CSP solver engine of the diagnosis algorithm is based on a public domain universal CSP solver library from the University of Atlanta [6]. Different backtracking and preprocessing (consistency) algorithms are implemented and can easily be applied.

The solver is able to maintain certain dynamic CSPs: the constraints themselves can be altered during the solution process but their number cannot. Initially only the 'fixed', syndrome-independent constraints are generated, the others are 'always-true' (the solver always works with complete constraint graphs so the still undefined constraints have to be 'always-true').

There are 20 variables in the CSP model; they represent the fault states of the processors with their data connections and the routing chips. The cardinality of the variable domains is 48. (A processor can be fault-free, faulty or dead and each of its 4 data connections can be live or broken, resulting in $3 \times 16 = 48$ possible states. The routing chips have 20 possible states but the solver requires equal domain sizes.)

Many sophisticated enhancements assure a maximal efficiency of the CSP solution. Only the variables/processors are considered that we have information about, so we get results only from the necessary units. Indistinguishable errors are merged into a single class by error collapsing. These modifications decrease both the number of variables processed and the number of value combinations checked and assure that only the valuable results are supplied. Further considerations can be adapted to the CSP solver very easily. For example if we consider a limited number of faulty units, the CSP solver can check whether this consideration holds and even automatically increases the error limit. This feature makes the system extremely fast for a few of errors and is still usable with errors of a number above the limit.

2.1.4 Results of a Test Run

The CSP-based diagnosis algorithm was tested with a logic fault injector: the host machine generated a random fault pattern for the Parsytec processors and downloaded it with the testing initialization messages. The low-level testing mechanism on the Parsytec processors interpreted the fault pattern and acted according to the fault state: 'fault-free' processors tested their neighbours and sent the results back to the host, 'faulty' processors did the testing but reported a random result and 'dead' processors did nothing. This construction was necessary because no physical fault injection was available for the Parsytec prototype equipped with T805 transputers and the fault injector developed for T9000 was unusable due to the difference in hardware structure.

Fig. 3 shows the results of a typical test run. In this case the fault pattern contained a single faulty processor. The upper curves display the number of the solutions found by the CSP solver and the number of processors that have already sent some test results and the lower ones display the number of consistency checks made as a measure for the computational efficiency.

2.2 The Distributed Approach

The centralized diagnosis algorithm described above is suitable for minor configurations of the Parsytec multiprocessor system because the use of at least one central resource (the host machine) is inevitable. Constraint-based system diagnosis, however, can be implemented in a fully distributed environment as well for large configurations, avoiding the time-consuming node-host communication for each individual syndrome part. An alternative algorithm was developed to evaluate this idea.

In this approach the diagnosis is made by each of the transputers individually, not by a central supervisor machine. Another important difference is that syndrome messages can be corrupted in much more interesting ways since they travel via a chain of transputers and links, not only one link.

2.2.1 The Fault Model

The fault model of this approach contains a lot of simplifications compared with the centralized one.

The system consists of processors and links forming a two-dimensional grid. There are no routers in this model.

The fault state of a processor can be either good or faulty (omission fault is considered). No distinction is made between faulty and dead processors, in contrary to the centralized approach, as in this approach the rapid reconfiguration using the spare processors is of primary priority in order to minimize the error-related computing time loss. 'Suspicious processors' are checked off-line.

In order to save communication bandwidth and to reduce the diagnostics related computational overhead, only elementary syndrome messages with a test outcome indicating a faulty unit will be sent by the testers to its neighbours and no message transfer is invoked in the other case.

A faulty tester processor will generate and distribute random syndrome elements, according to the PMC test invalidation model. In addition, syndrome messages from other processors will be potentially corrupted or simply not forwarded by a faulty processor. The majority of changes can

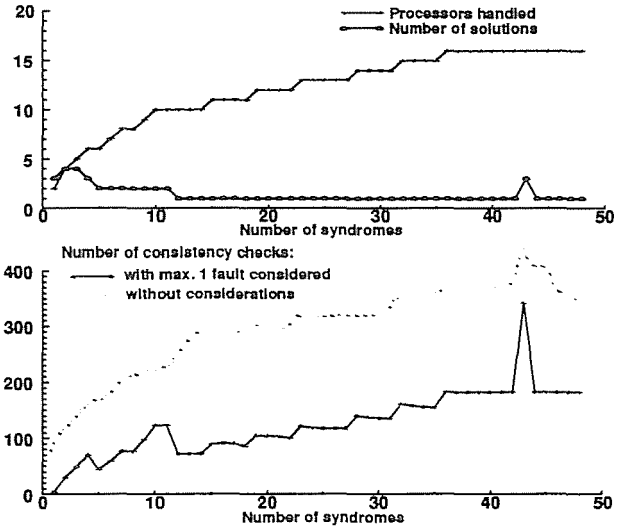

Fig. 3. Test results of the centralized CSP diagnosis algorithm

be detected using a simple error detecting protocol (e.g. checksum protection). The diagnostics program assumes that no undetected changes will occur. The second case of syndrome losses cannot be detected in such a simple way, therefore the diagnosing processor will never or only occasionally receive fault reports from unfortunately placed – good or faulty – processors [3].

The fault state of a link can be 'good' or 'faulty'. In the latter case the communication is blocked. A change in the contents of messages is impossible for the reasons stated above.

2.2.2 Representation of the Constraints

A variable is assigned to a processor and the links to its eastern and northern neighbours, thus for the 4×4 grid 16 variables are used. (the 0th bit to the eastern link, the 1st bit to the northern link and the 2nd bit to the processor). The domain consists of 5 elements (*Fig. 4*):

- FAULTFREE (000)
- FAULTEAST (001)
- FAULTNORTH (010)
- FAULTBOTH (011)
- FAULTPROC (100)

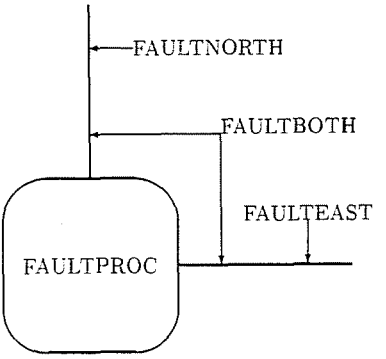

Fig. 4. Assignment of fault states to a variable

A part of the implication expressing the dominance of a processor fault over the link faults in the form of

$1_P \Rightarrow 0_{L_1} \wedge 0_{L_2} \wedge 0_{L_3} \wedge 0_{L_4}$ (the L_p indicates a link connected to **P**)

is already incorporated, i.e. the values 101, 110 and 111 are not allowed.

2.2.3 Reducing Communication Overhead

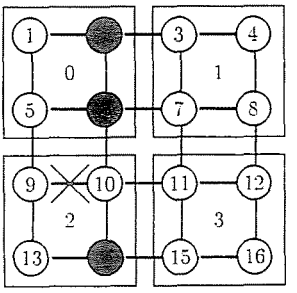

Fig. 5. Fault pattern in a system; the failure of link 9–10 is transient.

If simply the CSP based on the arriving syndromes indicating a fault were solved, this would produce – in most practical cases – a huge number of solutions from which practically no useful diagnosis can be generated. This problem originates in the small number of available syndrome elements, as typically only few components fail and only error messages are forwarded; at this point no information is available about processors which tested each

of their neighbours fault-free. Thus the solution space remains large due to the few constraints generated.

To overcome this problem, the set of components is determined which are undoubtedly fault-free, i.e. fault-free in each of the solutions.[1] Then it is possible to obtain which fault-free processors are reachable from the diagnosing one via fault-free links and processors. Tests show that this algorithm works only if a fault-free 'starting point' is given; that is the reason why the diagnosing processor is assumed to be fault-free. If it were faulty, one could not rely on the diagnosis made by it anyway.

These 'reachable' fault-free processors are important, because they allow to draw additional conclusions. They have the special property that all syndrome messages they generated were received by the diagnosing processor, therefore in case no messages arrived from one of them, one can conclude that *all tests performed by it had the outcome fault-free.* Thus new constraints can be extracted and consequently the solution space shrinks.

The constraint network should be solved repeatedly until no new constraints can be extracted.

It is important to note that this simplified strategy implicitly contains the assumption that links can have only permanent failures, i.e. the fault state of a link does not change during one diagnosis step. This assumption is justified by the short syndrome processing time. Without this assumption it cannot be ensured that the set of 'reachable' processors will be determined correctly. An example is shown in *Fig. 5*; if the faulty link behaves fault-free when processor #9 tests processor #10 but blocks communication during the distribution of messages, the diagnosis program comes to the conclusion that every processor is 'reachable', therefore the fact derived by the program that the link 1–2 is tested both faulty and fault-free leads to a global contradiction, which indicates that a new fault has occurred.

2.2.4 Permanent Failures

The program can be given some certified diagnosis results representing permanent faults in the system. These conditions often ease solving the CSP. Transient and permanent failures are distinguished in the following way: in the first phase diagnosis is based only on the permanent fault information, then diagnosis uses information from all faults; in the latter the failure type of each item is set to permanent in case the same item occurs in the former.

[1] The importance to consider a maximal number of faults should be clear: without it, even the solution 'every component is faulty' would always occur according to the applied PMC test invalidation model; hence the idea would fail.

2.2.5 Performance Tests of the Algorithms

The efficiency of the algorithm is strongly influenced by the preset upper limit for the number of faults, as the number of consistency checks increases fast for small limits (*Fig. 6*). The decision which limit is the most appropriate one depends on the failure characteristics of the system; a high limit increases running time unnecessarily while an excessively low limit can lead to contradiction and thus prohibiting diagnosis if there are more faults than this limit.

3. Conclusions

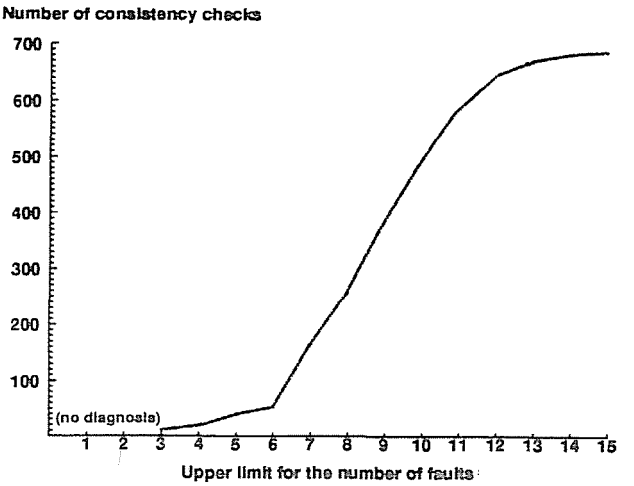

Fig. 6. Performance measures of the distributed algorithm

The CSP-based syndrome decoding algorithms have proved their proper operation during the tests and the correctness of the concepts behind it. Additional tests showed that the constraint solver is up to five times faster than an exhaustive search (the average processing time of a test result was 88 μs vs. 448 μs in a test series). However, the applied CSP solving algorithm (graph-based backjumping [6]) has not yet theoretically proved to be optimal for this application; further work is needed to find the most efficient strategy for the solver. The main advantages of the new approach presented are the following:

- It can be realized both in a centralized as in a distributed form as well.
- It supports the evaluation of partial information in the form of a diagnostics on-the-fly.
- More detailed and thus realistic functional fault models can be handled.
- Diagnosis in inhomogeneous systems can be performed as well.

References

1. BARBORAK, M. – MALEK, M. – DAHBURA, A.(1993): The Consensus Problem in Fault-Tolerant Computing. *ACM Computing Surveys*, Vol. 25, No. 2, pp. 171–220.
2. PATARICZA, A. – TILLY, K. –SELÉNYI, E. – DAL CIN, M. – PETRI, A.: Constraint-based System Level Diagnosis of Multiprocessor Architectures. *Proceedings of μP'94*, Budapest, Vol. I, pp. 75-84.
3. ALTMANN, J. – BARTHA, T. – PATARICZA, A. : On Integrating Error Detection into a Fault Diagnosis Algorithm for Massively Parallel Computers. *Proceedings of IEEE IPDS '95 Symposium*, Erlangen, Germany, pp. 154–164.
4. PETRI, A. Jr.: A Constraint-based Algorithm for System Level Diagnosis. Diploma Thesis, Technical University of Budapest, 1994.
5. PATARICZA, A. – TILLY, K. – SELÉNYI, E. – DAL CIN, M.: A Constraint Based Approach to System Level Diagnosis. Internal Report 4/1994., Friedrich-Alexander University, Erlangen-Nürnberg.
6. MANCHAK, D. – VAN BEEK, P.: A Binary CSP Solution Library for C Language. (Available by FTP from ftp.cs.ualberta.ca: /pub/ai/CSP).
7. SELÉNYI, E. (1986): System Level Fault Diagnosis in Multiprocessor Systems with a General Test-invalidation Model. Dr.Sc. Thesis, Hungarian Academy of Sciences.
8. TILLY, K.(1994): Constraint Based Logic Test Generation. Ph.D. Thesis, Hungarian Academy of Sciences.
9. MONTANARI, U.(1974): Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, Vol. 7, pp. 95–132.
10. MACKWORTH, A. – FREUDER, E. C.(1985): The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence*, Vol. 25, pp. 65–74.