Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki, i Telekomunikacji

KATEDRA INFORMATYKI

PRACA DOKTORSKA

ARKADIUSZ SZYMCZAK

WSPÓŁBIEŻNE SOLVERY ADAPTACYJNE DLA PROBLEMÓW INŻYNIERYJNYCH STEROWANE SIECIAMI PETRIEGO

PROMOTOR:
dr hab. Maciej Paszyński, prof. AGH

Kraków 2014

OŚWIADCZENIE AUTORA PRACY

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie, i nie korzystałem ze źródeł innych niż wymienione w pracy.

. . . .

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Computer Science, Electronics, and Telecommunications

DEPARTMENT OF COMPUTER SCIENCE

PH.D. THESIS

ARKADIUSZ SZYMCZAK

PETRI NETS CONTROLLED CONCURRENT ADAPTIVE SOLVERS FOR ENGINEERING PROBLEMS

SUPERVISOR:
Maciej Paszyński Ph.D, D.Sc.

Krakow 2014

I would like to express my true appreciation to Prof. Maciej Paszyński for his guidance, patience and great attitude that were of invaluable help to me in my research concluded with this thesis.

…

Table of contents

Chapter 1

Introduction

1.1. Motivation

The class of adaptive algorithms based on two- and three-dimensional computational meshes is widely utilized to solve many engineering problems in the domains such as nano-litography simulations (the process of production of micro-processors, so-called Step-and-Flash Imprint Lithography simulations) [50], propagation of electromagnetic waves (so-called Maxwell problem), in particular with oil-industry applications [40, 39, 37, 46], material science [21, 48, 47] as well as heat transfer problems [48, 47].

The aforementioned class of algorithms utilizes adaptive finite element method and other similar computational methods.

A formal model of adaptive algorithms, based on Petri nets and graph grammar, would allow to perform formal proofs of correctness of those algorithms. In particullar, it would allow for detection and prevention of deadlocks that could occur during adaptation of computational meshes.

In this thesis I focus on construction of Petri net models for particular implementation of adaptive algorithms, namely the two-dimensional self-adaptive hp finite element method code *hp2d* as described in [10] as well as three-dimensional self-adaptive hp finite element method *hp3d* code as described in [11]. I show that the original implementations of those algorithms may lead to a deadlock problem for certain configurations of mesh refinements. This happens during two-dimensional simulations of magnetolluric measurements process [60, 1] as well as during three-dimensional simulations of resistivity logging while drilling measurements [62, 40, 37]. Both problems are related to geolocation using antennas transmitting or receiving electromagnetic radiation, in order to localize oil and / or gas bearing formations in the ground.

The constructed Petri nets models allow to detect deadlock problems in both 2D and 3D problems. They also suggest the modification of 2D and 3D mesh adaptation algorithms, making them deadlock-free. Thus, the Petri nets models have been utilized for detection of deadlock problems in the original mesh adaptation algorithms as well as for proving that the new modified algorithms are actually deadlock-free.

Scientific results presented in this thesis are not limited to specific implementations of the *hp2d* and *hp3d* algorithms. They may be easly generalized to any other implementation of the mesh refinements algorithm. This is in particular true since *hp2d* and *hp3d* algorithms so far are the most complicated versions of the adaptive algorithms.

1.2. Main thesis and the structure of this book

The main thesis of this work may be formulated as follows:

It is possible to develop a formal model, based on Petri nets and graph grammar, for a class of adaptive algorithms based on two-dimensional and three-dimensional computational meshes, that allows for formal analysis of the correctness of these algorithms, in particular for detection of the deadlock problem.

The rest of the book is organized as follows:

– I start with presenting the state-of-the-art in the field of adaptive algorithms for finite element method and graph grammar models.
– I introduce several deadlock problems that occured during two- and three-dimensional computations with adaptive finite element method.
– I summarize the list of open problems related to correctness analysis of mesh refinement algorithms and detection of the deadlock problem.
– I summarize the main scientific results obtained in this thesis.
– In the following sections I introduce formal definitions of the Petri net model.
– The adaptive algorithm as implemented in *hp2d* code [10] is expressed by graph grammar productions.
– Petri net model for deadlock detection in anisotropic mesh adaptation algorithm *hp2d* is presented.
– The adaptive algorithm *hp2d* is corrected by including some additional graph grammar productions.
– The new Petri net model for the enhanced graph grammar is created.
– The new Petri net model is proven to be deadlock-free, thus the new adaptive algorithm *hp2d* is proven to be deadlock-free.
– The entire consideration is repeated for three-dimensional adaptive computations.
– The adaptive algorithm as implemented in *hp3d* code [11] is expressed by graph grammar productions.
– Petri net model for deadlock detection in anisotropic mesh adaptation algorithm *hp3d* is presented.
– The adaptive algorithm *hp3d* is corrected by including some additional graph grammar productions.
– The new Petri net model for the enhanced graph grammar is created.
– The new Petri net model is proven to be deadlock-free, thus the new adaptive algorithm *hp3d* is proven to be deadlock-free.
– Conclusions on the research results and remarks for future research are discussed.

1.3. State-of-the-art

1.3.1. Adaptation algorithms

This section provides a short presentation on the current state-of-the-art in the field of the development of sequential and parallel mesh refinement algorithms for h adaptive grids. The adaptive algorithms can be classified in the following way:

– *Uniform h adaptation*: all finite elements are uniformly broken into smaller elements.

– *Uniform p adaptation*: the polynomial order of approximation is increased uniformly over the entire mesh, e.g. by adding bubble shape functions of higher orders over element edges and interiors.

– *Non-uniform h adaptation*: some finite elements are broken into smaller elements, only in those parts of the mesh which have high numerical error.

– *Non-uniform hp adaptation*: some finite elements are broken into smaller elements, and the polynomial orders of approximation are increased, only in those parts of the mesh which have high numerical error.

– *r adaptation* where the mesh is re-generated using new distribution of elements.

For non-uniform h or hp adaptation, it is necessary to locate finite elements with high numerical error. In the case of non-uniform hp adaptation, it is also necessary to select the optimal refinements for such finite elements. The non-uniform h or hp adaptation process can be executed in the following ways:

– The selection of the finite elements to be refined and the type of refinement depends on the user.

– The selection of the finite elements to be refined and the type of refinement depends on an algorithm based on the knowledge of the structure of the solution.

– The selection of the finite elements to be refined and the type of refinement depends on the self-adaptive algorithm, which is designed without any particular knowledge of the structure of the solution, and works in fully automatic mode, without any user interaction.

The first and the second algorithm are reffered to as the non-automatic adaptation, whereas the third algorithm is called automatic adaptation. In particular, the non-uniform hp automatic adaptation is called the self-adaptive hp Finite Element Method (self-adaptive hp-FEM). The r adaptation is also often referred to as *re-meshing*.

Algorithms for the uniform h, uniform p, non-uniform h and non-uniform hp automatic adaptation for 3D grids have been designed, implemented and tested by the group of prof. Leszek Demkowicz [11]. Many authors followed the approach originated by Demkowicz and implemented their own variations of these algorithms. In [33], authors employ modern h and hp adaptation algorithms for the Girkmann problem. [3] presents the h adaptation approach using the octree data structure and the ideas originally introduced by [11] for local h refinements. The uniform h adaptation algorithm has also been utilized for the solution of the projection problem [23].

The r-adaptation is also commonly used in the computational community. Paper [29] uses the re-meshing algorithm for modeling large deformations in geological problems. The r adaptation algorithm can also be utilized for solution of non-stationary problems. An example of that is modeling of the wind flow around a bridge, as presented in [52].

The self-adaptive h-FEM or hp-FEM algorithm may utilize different error estimators for guiding the adaptation process. There are different error estimators defined for elliptic [4, 7], parabolic [19, 8] or multi-physics problems [34]. From the point of view of deadlock modeling, the error estimator does not influence the problem.

This dissertation focuses on modeling deadlock scenarios in the self-adaptive h-FEM algorithm. The addition of automatic p adaptivity is also possible, since playing with different polynomial orders of approximation over the edge does not influence the deadlock problem, which results from h adaptation only. The Petri net model presented in this work can be also applied to model the other non-automatic adaptation algorithms. The only exception is the r adaptivity.

The parallelization of any of the above adaptation algorithm results in distribution of the computational mesh over a number of processors. Among major undertakings to develop the algorithms supporting h refinements over the computational mesh divided into sub-domains, first of all one has to list the Sierra Environment [15, 16, 17, 18]. There are also some ongoing object-oriented projects developing distributed data structure hp-adaptive FEM for flow simulations [2]. The original hp-adaptive algorithm from [11] has also been parallelized using either domain decomposition approach [47] or OpenMP approach [53, 54, 55]. Other parallel hp adaptive algorithms implemented so far have been developed by [14, 56] in the context of Discontinuous Galerkin (DG) methods. The only parallel hp adaptive algorithms for Continuous Galerkin method that I am aware of have been developed by [51, 31]. None of these algorithms supports automatic hp adaptation.

The model presented in this thesis can still be applied for analyzing the deadlock problem for parallel mesh refinements algorithm, assuming the deadlock scenario takes place within a single sub-domain, or the sub-domains have been collected into a single processor.

1.3.2. Mesh adaptation and deadlock problem

This section introduces the deadlock problem encountered in two- and three-dimensional mesh adaptation algorithms.

Isotropic mesh refinements in 2D break selected finite elements in two directions to construct four son elements. Anisotropic refinements break selected elements in one or two directions, producing two or four son elements, respectively. Isotropic mesh refinements in 3D break selected finite elements in three directions to construct eight son elements. Anisotropic refinements break selected elements in one, two or three directions, producing two, four or eight son elements, respectively. Let us consider anisotropic mesh refinement algorithms for 2D grids composed of rectangular finite elements as presented in [10], and for 3D grids composed of hexahedral finite elements as presented in [11]. Incorrect implementation of the anisotropic h-adaptation algorithm may result in a deadlock scenario, where some requested refinements are impossible to execute.

To make the implementation of local refinements tractable while ensuring continuity in the finite element solution, many anisotropic refinement codes support the so-called *1-irregularity rule*. According to that rule, an element with hanging nodes (i.e., a node whose coefficient value is determined by neighboring nodes in order to impose global continuity of the solution), cannot be further refined. For refining such an element, one needs to refine one or several neighboring elements first in order to eliminate all hanging nodes.

The 1-irregularity rule is presented in Figure 1.1 for the 2D case. The rule prevents unbroken element edges from being adjacent to more than two finite elements on one side. When an unbroken edge is adjacent to one large finite element on one side and two smaller finite elements on the other side, the approximation over these two smaller elements is constrained by the approximation over the larger element. This is illustrated in Figure 1.2. To avoid technical nightmare with constrained approximation over multiple constrained edges, the 1-irregularity rule is commonly used in the h adaptive algorithms. To satisfy the 1-irregularity rule, several additional refinements on large adjacent elements may be required. As it is presented in this thesis, the 1-irregularity rule may lead to a deadlock problem if the mesh adaptation algorithm is not designed correctly.

This problem is also illustrated in Figure 1.3 for the 3D case. Let us consider two finite elements, one broken into eight son elements, and the other one unbroken. In the finite element method nomenclature, the nodes and vertexes of the shared face are called constrained, in the sense that the approximation over four small faces of the small elements is constrained by the approximation over one big face of the big element.

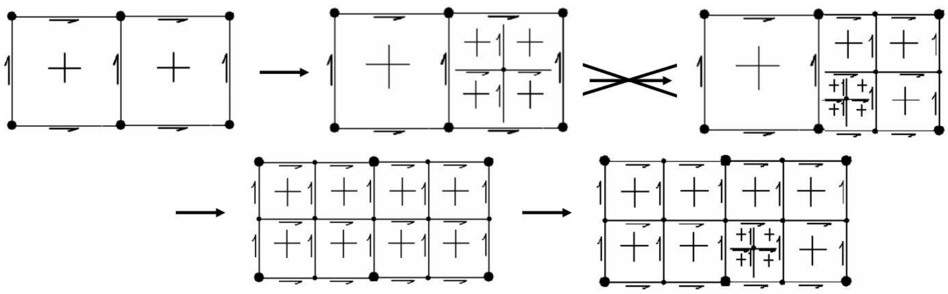


Figure 1.1: 1-irregularity rule: a finite element can be broken only once without breaking the adjacent large elements.

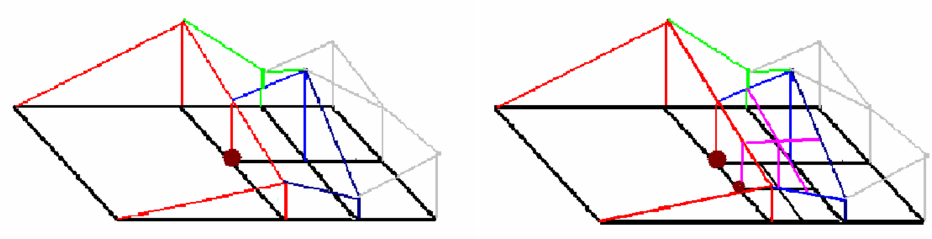


Figure 1.2: 1-irregularity rule: approximation over the common edge is constrained by the big element.

If we break one of the small elements for the second time, the resulting state is forbidden, compare Figure 1.4. The reason is that in such a case, we will have double constrained nodes over the smallest faces of the small broken element. To prevent such forbidden state, it is necessary to break the large neighbor before breaking the small element, compare Figure 1.5.

As it has been experienced during numerical experiments, the 1-irregularity rule has one unexpected drawback, both in two and three dimensions. Namely, it may result in a deadlock of the adaptation process. It should be emphasized that the mesh at deadlock state is at an acceptable state, the numerical problem can be solved on that mesh, however further refinements are not possible here.

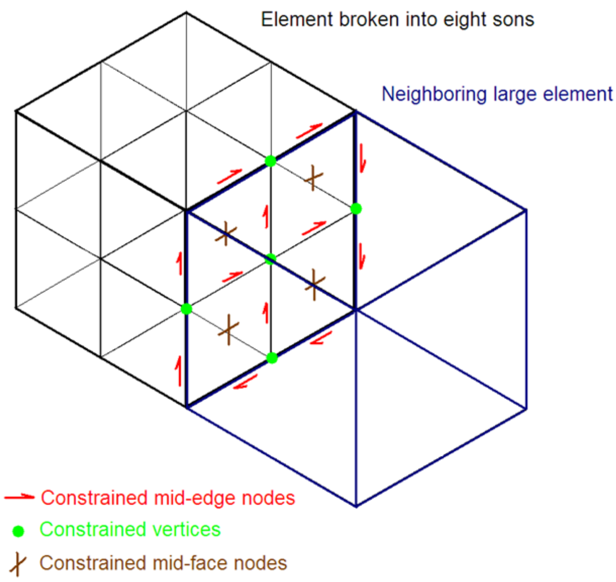


Figure 1.3: Two adjacent elements, one broken into eight son elements.

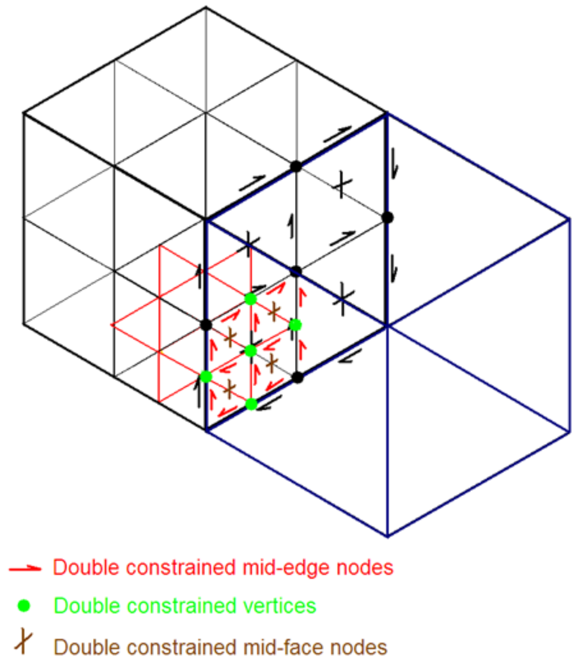


Figure 1.4: The forbidden state with double constrained nodes.

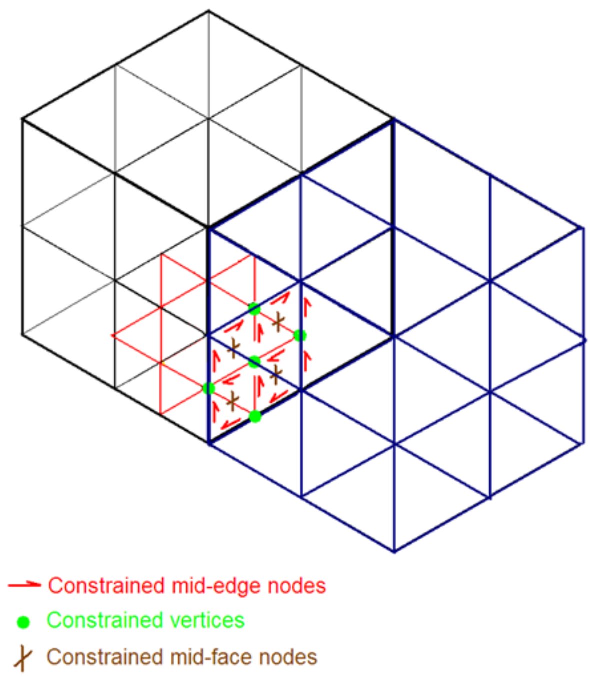


Figure 1.5: Breaking large adjacent element followed by breaking one of the small elements. No double constrained nodes.

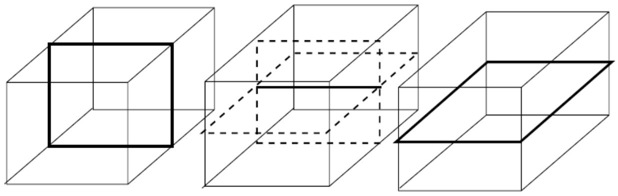


Figure 1.6: A first deadlock scenario.

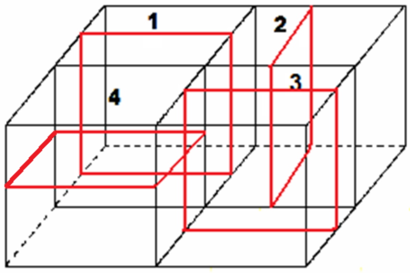


Figure 1.7: A second deadlock scenario.

Practical motivation for this work was the communication with prof. David Pardo, working on 3D anisotropic mesh refinement algorithm used for the simulations of 3D DC resistivity logging measurements in deviated wells [38]. This problem is of great interest to the geophysical community [32, 35, 22]. During those computations, the deadlock problem occurred.

The deadlock problems are related to the 1-irregularity rule mentioned above. This rule implies that an edge of an element can be adjacent to no more than two smaller edges. Additionally, a face of an element can be adjacent to either two smaller faces, or to four smaller faces, provided they are broken in both directions. In the adaptive community, it is often said that the small edge is constrained by the big edge, or the small face is constrained by the big face.

One of the first deadlock problems in 3D h-adaptive computations with hexahedral elements was identified in [12]. The deadlock in that version of the adaptive code was caused by the fact that elements could be broken only into two son elements, either along **X** or **Y** or **Z** direction. In that paper [12] Figure 13 (reproduced here as Figure 1.6 with authors' agreement) illustrates a basic deadlock scenario. There are three elements in a row, the first one and the third one are broken in two different directions. Another request to break the first element implies the necessity of breaking the central element. This is because the left side face of the central element cannot be adjacent to twice-broken faces of the first element. If we break the central element in two directions, we will block the possibility of breaking the third element, since the central element would have to be broken in another direction to make breaking the third element possible. This deadlock problem was overcome by adding the possibility of breaking elements into four or eight son elements at the same time [12].

Actually, the authors of [12] found out that this additional breaking of elements should be performed into eight son nodes, to block unwanted propagation of additional refinements.

Another deadlock scenario, presented in Figure 1.7, has been identified. In this case, there is a patch of four elements. Each of those elements has been broken into two son elements. The resulting mesh does not violate the 1-irregularity rule, however, further refinements are not possible in this patch.

For example, let us assume that we want to break element 1 again in the direction perpendicular to the **X** axis. The right face of element 1 is constrained by the left face of element 2. We need to break element 2 first, into eight son nodes, in order to prevent further propagation of refinements. But we cannot do that, since the front face of element 2 is constrained by the rear face of element 3. We need to break element 3 into eight son nodes, but we

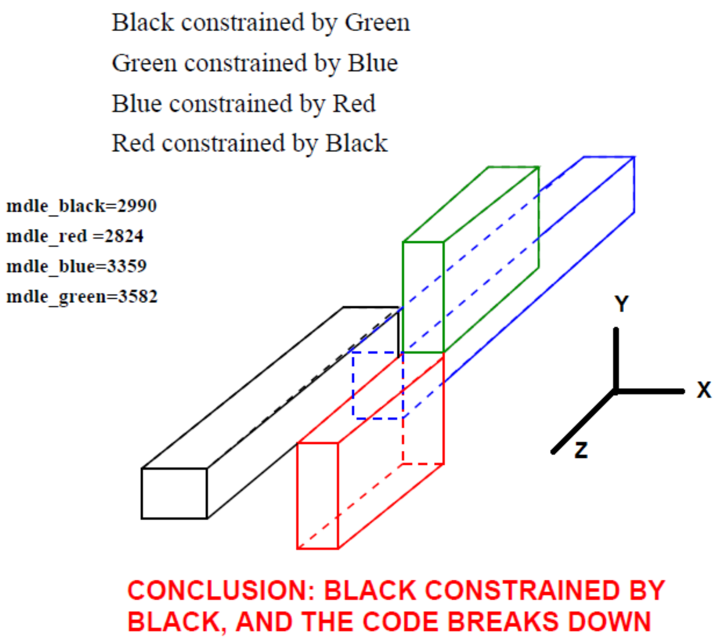


Figure 1.8: A third deadlock scenario.

cannot do that since the left face of element 3 is constrained by the right face of element 4. So we need to break element 4 into eight son nodes, but we cannot do that yet, since the rear face of element 4 is constrained by the front face of element 1. We have to break element 1 first, in the way that is contradictory to its original request for refinement. We have a deadlock scenario here.

Yet another deadlock problem has been found, much more complicated. In the 3D mesh described in Figure 1.8, there are four elements that touch one another through edges. We would like to break the black element one more time into four son nodes, along **Z** axis. This refinement request implies the necessity of breaking the green element, since the edge of the black element parallel to **Y** axis is constrained by the edge of the green element parallel to **Y** axis. We want to break the green element into eight son nodes, to prevent unwanted propagation of refinements. But this is not possible yet, since the **Z** edge of the green element is constrained by the **Z** edge of the blue element. The blue element also must be broken into eight son elements. But again, this is not possible, since **Y** edge of the blue element is constrained by **Y** edge of the red element. In turn, the Z edge of the red element is constrained by the **Z** edge of the black element. Again, we encounter a deadlock scenario here.

Other deadlock scenarios have been also reported in non-structural 3D tetrahedral adaptive finite element method computations [30].

In this work I propose the use of a Petri net for detecting deadlock scenarios. I will show how to overcome the deadlock problem, and how to prove formally by using reachability graphs of Petri nets that the proposed corrected mesh transformations are actually deadlock-free.

Modeling of deadlock scenarios for adaptive finite element methods was already performed for two dimensional (2D) anisotropic refinements of rectangular meshes. In the first attempt [59], only a 2D sub-mesh with 2×2 rectangular elements was modeled, and a Petri net was constructed for detecting deadlock scenarios in such a simple example. This result is generalized for arbitrary 2D rectangular grids in this dissertation. Please also refer to [60]. Subsequently, in [45] a Petri net model was designed for a 3D sub-mesh with $2 \times 2 \times 2$ hexahedral elements, and it was shown how to prevent deadlocks in such an example. This thesis generalizes those results to the class of arbitrary 3D hexahedral meshes in a similar way as it was performed in [60] for the case of 2D rectangular grids [59]. Please also refer to [62].

The Petri net model is independent of the numerical problem being solved, however it depends on the particular mesh adaptation algorithm. The Petri net models described in this thesis have been implemented in PIPE software [9]. Reachability graph construction and the deadlock analysis have been executed by using automatic tools implemented also in PIPE software. Once we have a corrected version of the adaptation algorithm proven to be deadlock-free, this algorithm can be used to solve any numerical problem without incurring a deadlock.

1.3.3. Graph grammar models

Representation of the topological structure of computational meshes as a hierarchy of vertices, edges, faces and regions has been proposed by [6]. The first attempt to model mesh transformations by graph grammar productions has been proposed by [20], for regular triangular two-dimensional meshes with h adaptation. The authors utilized a quasi context sensitive graph grammar there. However, the application of the quasi-context sensitive graph grammar for hp- adaptive mesh transformation seems to be limited, because the mesh transformations, such as the enforcement of 1-irregularity rule or the minimum rule, are context dependent and cannot be modeled by a quasi-context sensitive graph grammar. In [57] a simple topological chain rewriting framework modeling mesh refinement process is described. The approach presented there also allows for modeling uniform mesh refinements only. Recently, the Composite Programmable Graph Grammar (CP-GG) for modeling the mesh refinements as graph grammar productions (graph transformations), in both 2D [43, 44, 61] and 3D [41, 42] has been introduced. The CP-GG, which has been introduced in [25], is a generative tool for the design process using graph transformations executed on a graph representation of the designed object [24, 26, 27].

Expressing mesh transformations by graph grammar productions simplifies construction of the Petri net model. Thus, this work introduces several graph grammar productions representing anisotropic refinements of either 2D or 3D computational meshes with rectangular and hexahedral elements. Graph grammar productions presented in this work are introduced using a simplified notation, for both 2D and 3D meshes. This is because the details of the CP-GG model are not part of this dissertation. All is needed is a high level abstract notation for expressing mesh transformations in order to build Petri nets on top of them. For the formal definitions please refer to [62].

1.4. Open problems

This section summarizes the open problems related to the field of correctness analysis of mesh adaptation algorithms.

1. There is no model allowing for formal correctness analysis of two-dimensional mesh adaptation algorithms, like *hp2d* algorithm presented in [10].
2. There is no model allowing for formal correctness analysis of three-dimensional mesh adaptation algorithms, like *hp3d* algorithm presented in [11].
3. There is no model allowing for automatic detection of potential deadlock during execution of two-dimensional mesh adaptation algorithms, like the one presented in [10].
4. There is no model allowing for automatic detection of potential deadlock during execution of three-dimensional mesh adaptation algorithms, like the one presented in [11].
5. Deadlock problems have been reported during two-dimensional adaptive finite element method computations simulating the process of magnetolluric measurements for the detection of the oil / gas bearing formations in the ground [60, 1] .
6. Deadlock problems have been reported during three-dimensional adaptive finite element method computations simulating the process of resistivity logging while driling measurements for the detection of the oil / gas bearing formations in the ground [62, 40, 37]. .

1.5. Main scientific results

This section summarizes my main scientific results related to the open problems listed in the previous section.

1. Expressing the two-dimensional mesh adaptation algorithm *hp2d* as implemented in [10] by a set of graph grammar productions.
2. Development of a Petri net based model for the two-dimensional mesh adaptation algorithm *hp2d* as implemented in [10].
3. Detection of the deadlock problem for the two-dimensional mesh adaptation algorithm *hp2d* as implemented in [10].
4. Introduction of some additional graph grammar productions to the set of productions expressing the two-dimensional mesh adaptation algorithm *hp2d* as implemented in [10].
5. Proof for the deadlock-free enhanced graph grammar model expressing the two-dimensional mesh adaptations by using the Petri net model.
6. Correction of the two-dimensional mesh adaptation algorithm according to the enhanced deadlock-free graph grammar model.
7. Deadlock-free numerical results for two-dimensional magnetotelluric problem.
8. Expressing the three-dimensional mesh adaptation algorithm *hp3d* as implemented in [11] by a set of graph grammar productions.
9. Development of a Petri net based model for three-dimensional mesh adaptation algorithm *hp3d* as implemented in [11].
10. Detection of the deadlock problem for the three-dimensional mesh adaptation algorithm *hp3d* as implemented in [11].

11. Introduction of some additional graph grammar productions to the set of productions expressing the three-dimensional mesh adaptation algorithm *hp3d* as implemented in [11].
12. Proof for the deadlock-free enhanced graph grammar model expressing the three-dimensional mesh adaptations by using the Petri net model.
13. Correction of the three-dimensional mesh adaptation algorithm according to the enhanced deadlock-free graph grammar model.
14. Deadlock-free numerical results for three-dimensional direct current borehole resistivity measurement simulations in deviated wells.

These results solve the open problems listed in section 1.4.

Chapter 2

Petri net models of adaptive meshes

2.1. Petri net based detecting of deadlock during anisotropic adaptation of 2D rectangular meshes

2.1.1. Mesh adaptation algorithm

The mesh adaptation algorithm as implemented in *hp2d* code [10], can be summarized in the following way:

Algorithm 2.1.1. *(Automatic two-dimensinal mesh adaptations)*

```
1    L = List of elements el to be refined with refinement kind
2    do while  L not empty
3      el = get next element from L and its refinement kind
4      loop through edge ∈ edges of element el
5        if edge belongs to big neighbor element then
6          Store neighbor with its refinement kind at the end of L
7          Store el and its refinement kind at the end of L
8          continue do-loop from line 2
9        endif
10     enddo
11     break element el in a way kind using the virtual refinement
12   end while
```

Remark 2.1.1. *A* **deadlock** *scenario occurs when a virtual refinement propagates onto some adjacent element through an edge, and it is contradictory to the virtual refinement that has already been selected for the adjacent element.*

In the mesh adaptation Algorithm 2.1.1, the deadlock happens in line 11 when we try to break the interior of an element that has already been broken in some other way.

2.1.2. Graph grammar model

The mesh transformations performed by the mesh adaptation algorithm can be expressed by several basic tasks, called graph grammar productions. These tasks are summarized in Figure 2.1.

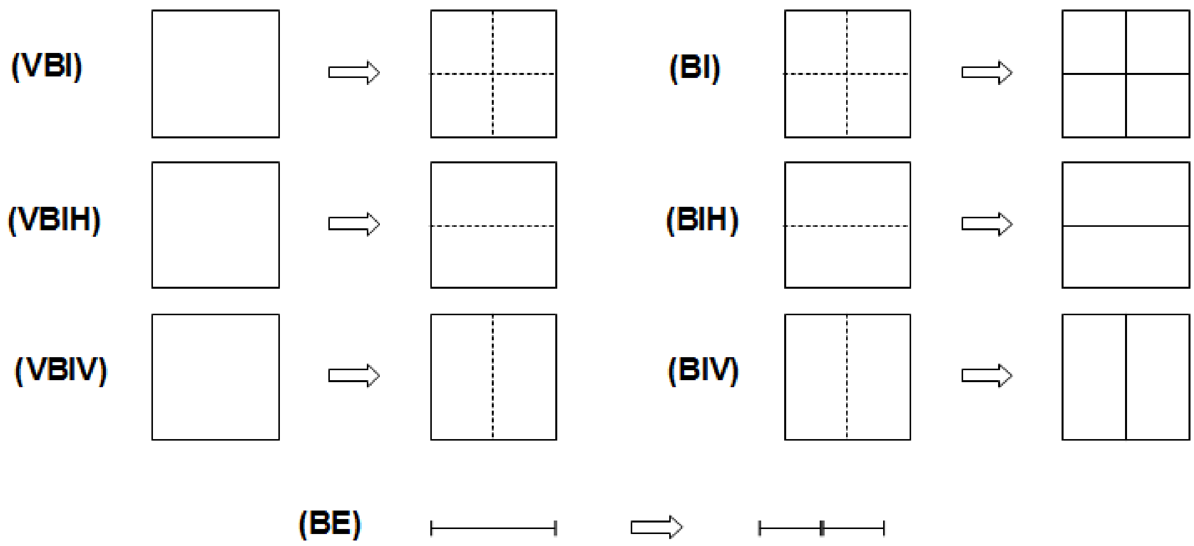


Figure 2.1: Graph grammar modeling the two-dimensional mesh adaptation according to *hp2d*.

Productions whose names start with **V** denote the so-called virtual refinements - requests for breaking an element interior in one of several possible directions. Productions **(VBIH)** and **(VBIV)** denote virtual breaking of an element interior in a single direction along X, and Y axis, respectively. Production **(VBI)** denotes virtual breaking of an element interior along two designated axis at the same time.

The computational mesh after execution of any of these virtual refinements is not in a legal state. This is because the virtual refinements break only the element interiors. Afterwards the mesh must be closed by enforcing additional breaking of some edges. An edge must be broken if it is surrounded by two broken interiors. Corresponding graph grammar productions for actual refinements are denoted by **B** letter. Let us assume that closing of the refinement process for edges can be expressed by one production **B*** whose name corresponds to the virtual refinement executed before.

According to 1-irregularity rule, a finite element edge can be broken only once. In other words, when we try to break a finite element edge for the second time, it is necessary to break large adjacent element first. It is assumed that the large adjacent element is always broken in both directions, to prevent long propagation of refinements.

Observation 2.1.1. *There are eight possible configurations when 1-irregularity rule implies breaking the large adjacent element, as shown in Figure 2.2.*

Observation 2.1.2. *The request to break the second large adjacent element occurs only when the first large adjacent element was already broken in the direction perpendicular to the direction of refinement propagation. The direction of refinement propagation* turns *each time it reaches a large adjacent element already broken in the direction perpendicular to the direction of refinement propagation.*

Proof. From the eight configurations considered in Observation 2.1.1 there are four configurations when such a requirement applies, as shown in Figure 2.3. In all of them the first large adjacent element is already broken in the direction perpendicular to the direction of refinement propagation. Notice that the large adjacent element to be broken may be also a half of some larger element, as shown in Figure 2.4. □

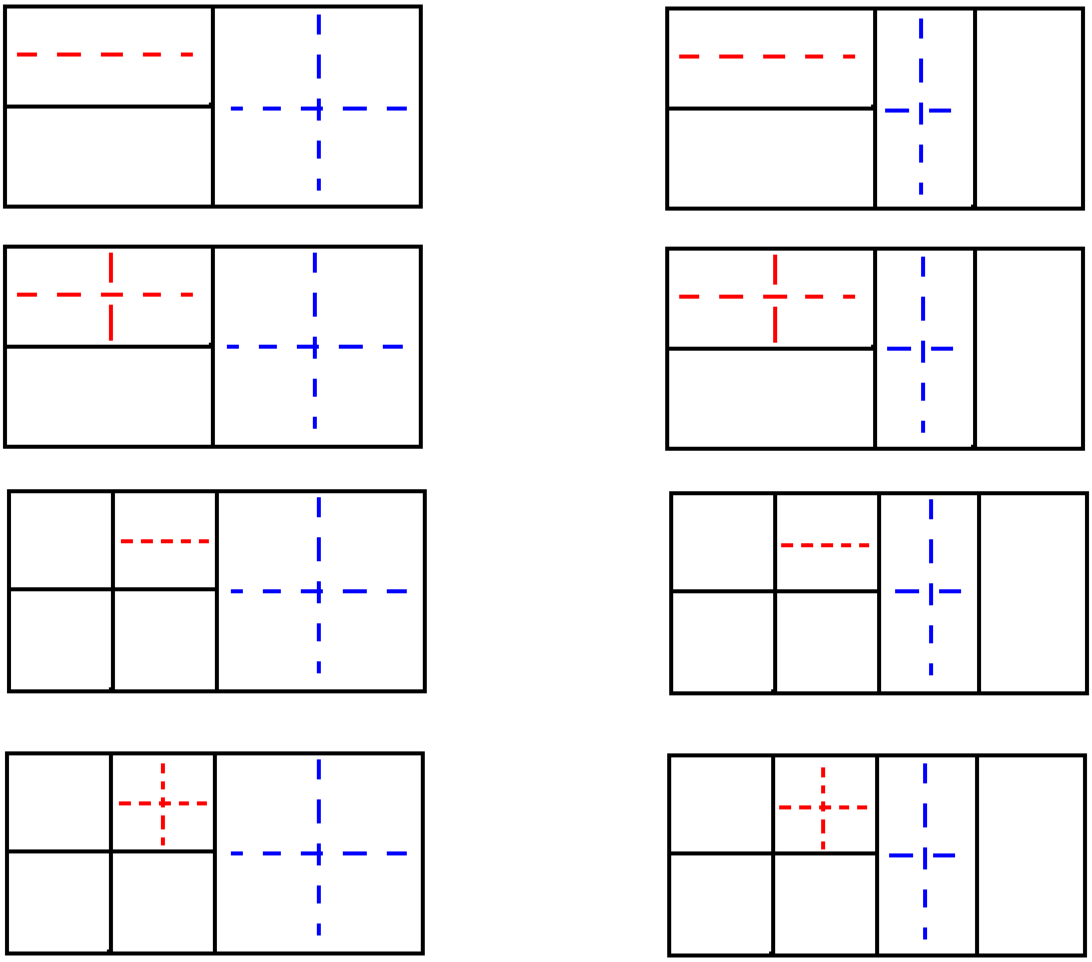


Figure 2.2: 1-irregularity rule configurations

2.1.3. Hierarchical Petri net model

Remark 2.1.2. *In order to detect a possible deadlock scenario, we need to construct a hierarchical Petri net with the main page covering the entire two-dimensional mesh, and sub-pages corresponding to pairs of elements adjacent through an edge.*

Let us construct the hierarchical Petri net model for the entire mesh, with main page corresponding to the entire two-dimensional mesh, and with sub-pages corresponding to all element pairs adjacent by an edge. Because of the 1-irregularity rule enforced over the entire mesh, all the pairs of elements are at the same level of adaptation. Each sub-page corresponding to a single pair of elements considers refinement request for any of the elements in the pair, with possible propagation to the other element in the pair. The sub-page considers also all possible refinement requests coming from the external elements. Thus, let us consider all possible combinations of two virtual refinement requests, and check if they can result in a deadlock.

The hierarchical Petri net model has been constructed in such a way that actual deadlock detection is performed by the sub-pages covering two-element patches of the two-dimensional mesh. The hierarchical Petri net sub-page for finite elements adjacent along X axis is depicted in Figure 2.5. The hierarchical Petri net sub-page for finite elements adjacent along Y axis is depicted in Figure 2.6.

The Petri nets described in this thesis are defined as hierarchical colored Petri nets (compare [58] p. 177 definition 10.16), limited to just one color (there is just one type of token). Sub-pages are linked to the main page

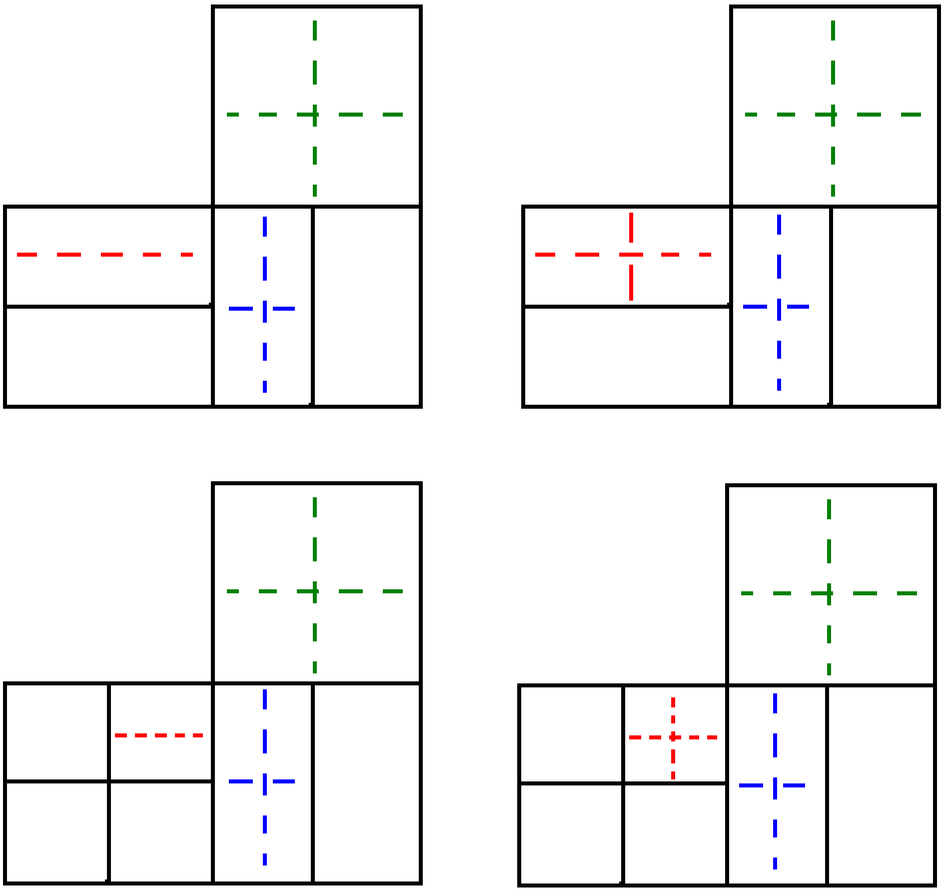


Figure 2.3: Double refinement propagations

by socket and port nodes. A socket is a place in the main page of the hierarchical Petri net that is linked to a port node in a sub-page. Such a pair of places behaves as a single common place (or a fusion) shared between the two pages (compare [58] p. 176).

The hierarchical Petri net subpage contains two starting places (**P0** and **P1**) - one for each mesh element of the modeled pair. The two upper rows of Petri net transitions are named after grammar productions they represent. Numbers at the end of transition names denote the corresponding mesh element a given transition pertains to. Firing any of those transitions models executing a corresponding grammar production. Remaining Petri net transitions model the following mesh element transformations:

- **VHA** - request (virtual) to break sub-element adjacent to the other element in the pair, along X axis;
- **VVA** - request (virtual) to break sub-element adjacent to the other element in the pair, along Y axis;
- **VA** - transformation breaking sub-element adjacent to the other element in the pair, along Y axis;
- **HA** - transformation breaking sub-element adjacent to the other element in the pair, along X axis;
- **VP** - transformation propagating the refinement request (virtual) onto the other element in the pair;
- **P** - transformation executing the propagated refinement.

The Petri net arcs define all possible execution paths for a round of mesh element refinements by enforcing dependency relationships between relevant transitions (productions). Whenever only one of a set of grammar productions can be executed, corresponding Petri net transitions depend on a common place with single token. Whenever exe-

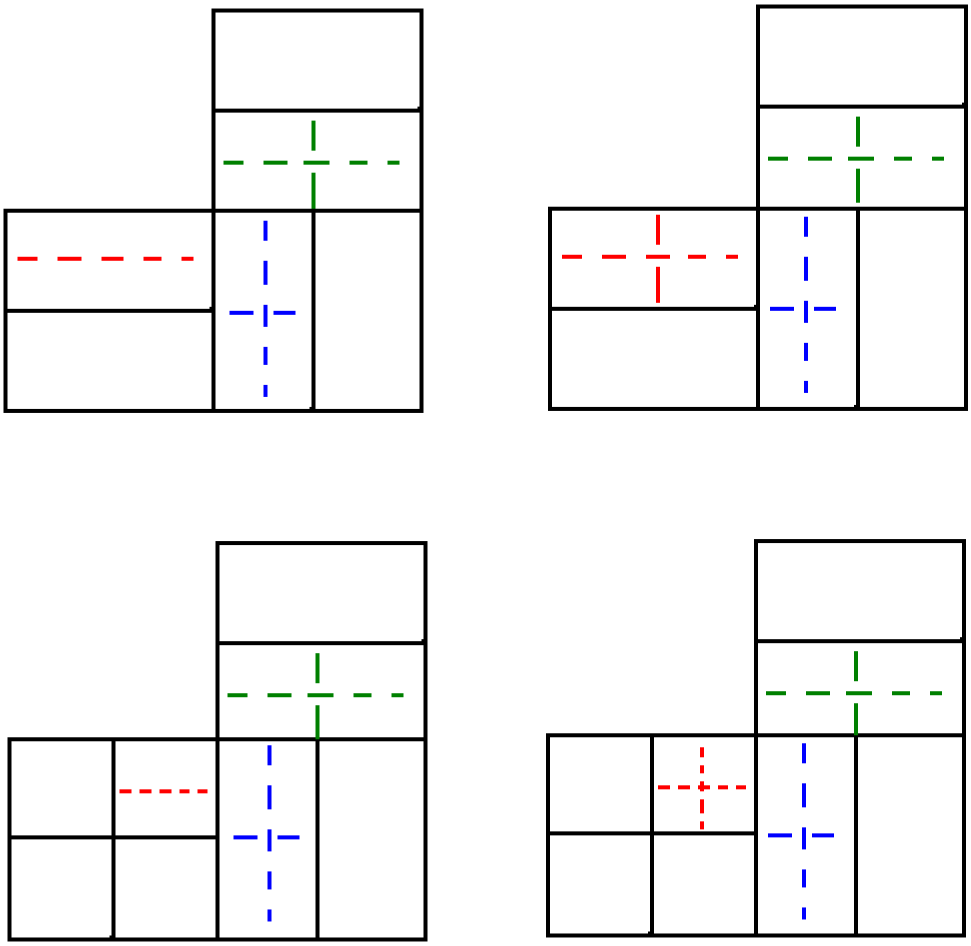


Figure 2.4: Double propagations to half-element

cution of a production blocks execution of another production, an inhibitor arc is used between corresponding Petri net transitions (actually between intermediate place and dependent transition).

Remark 2.1.3. *If the request is for the refinement in both directions at the same time, the potential propagation of refinement request cannot lead to a deadlock since the propagated requests are always for bidirectional refinements, hence such a propagated refinement request will never contradict the original bidirectional refinement request. That is why the bidirectional refinement requests for the sub-elements are omitted in the Petri net model.*

It is critical that each pair of adjacent mesh elements is covered with a Petri net subpage of appropriate type. To this end the algorithm of hierarchical Petri net generation for given finite element mesh has been developed.

Algorithm 2.1.2. *(generation of hierarchical Petri net for given FE mesh)*

1. *Create the main page of the hierarchical Petri net.*
    - *(a) Create a Petri net place for each element of the mesh being analyzed.*
    - *(b) For each pair of adjacent mesh elements create a Petri net transition and connect the corresponding places to this transition with arcs.*
    - *(c) Create an output place for each transition and connect each place to its corresponding transition with an arc.*
2. *Bind the main page of the hierarchical Petri net with the subpages.*

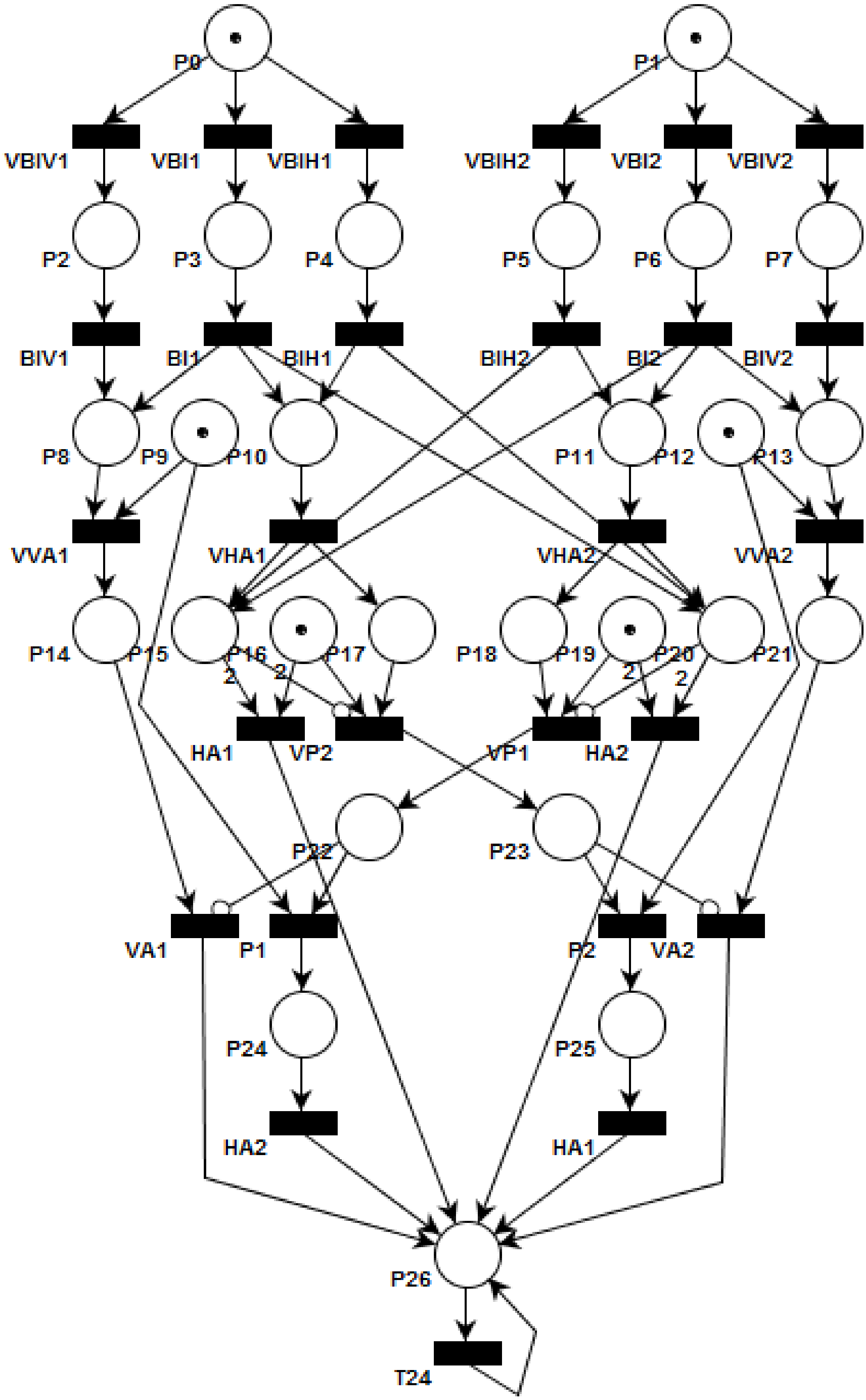


Figure 2.5: Hierarchical Petri net subpage for pair of mesh elements adjacent along X axis.

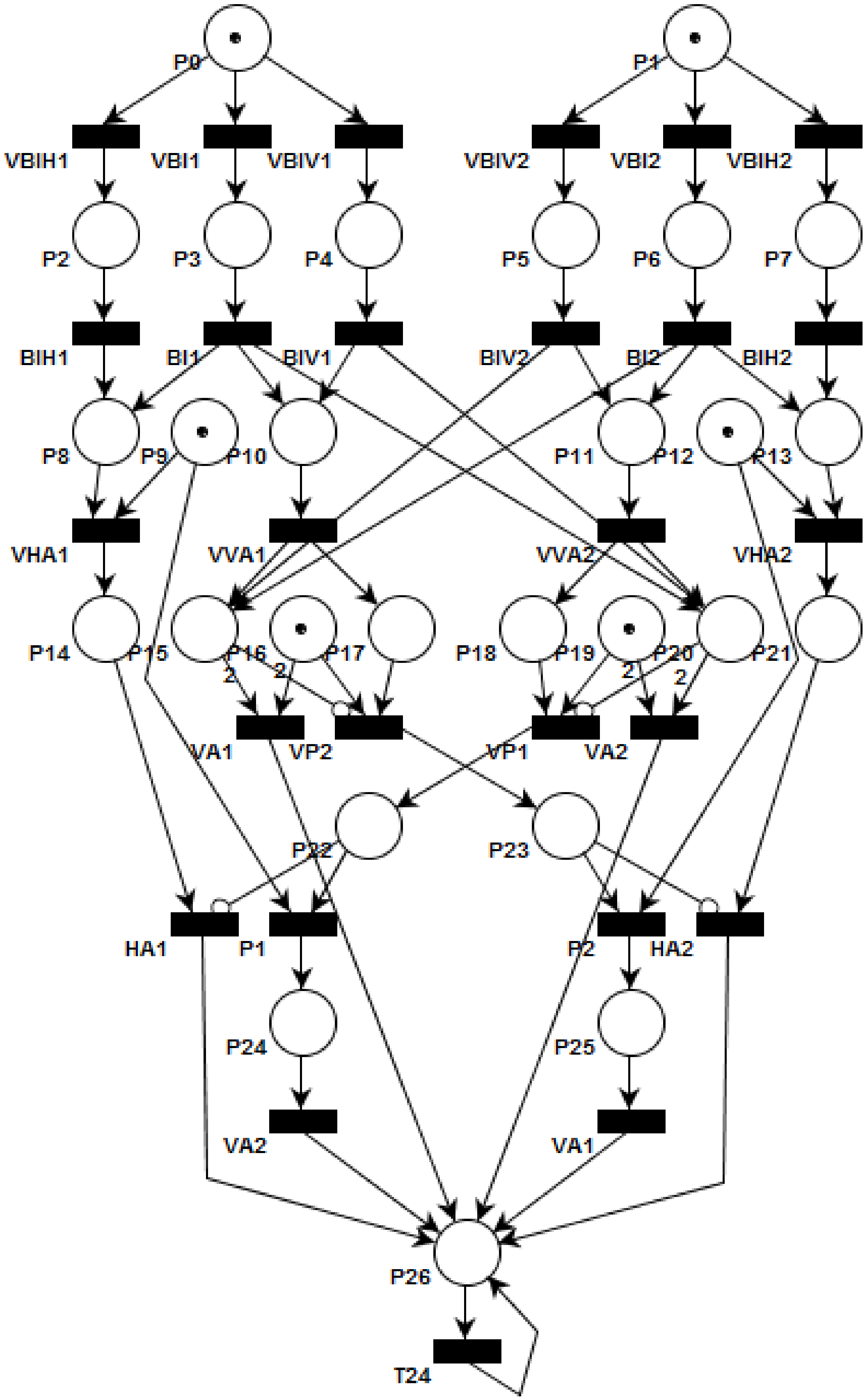


Figure 2.6: Hierarchical Petri net subpage for pair of mesh elements adjacent along Y axis.

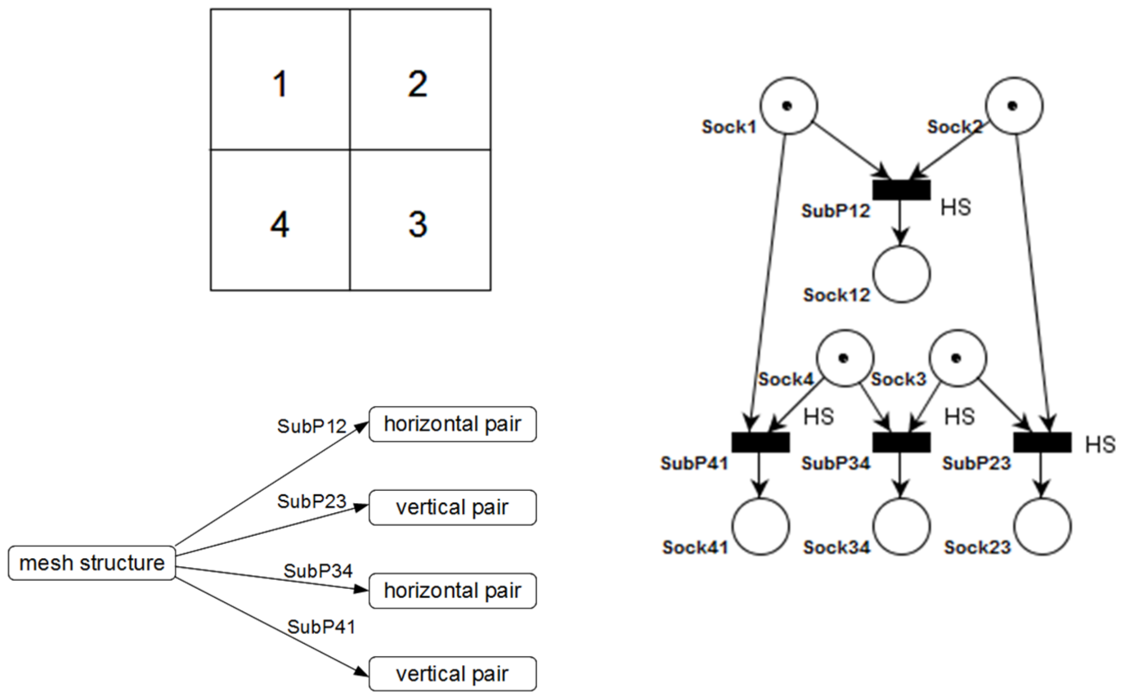


Figure 2.7: Main page of hierarchical Petri net for four finite element mesh

(a) *Substitute each transition in the main page with an appropriate subpage instance depending on whether the input places to the given transition represent mesh elements adjacent along X or Y axis.*

(b) *The input places of each transition in the main page become the socket nodes to the substituted subpage instance and are bound to the port nodes (places P0 and P1) in the substituted subpage instance.*

(c) *The output place of each transition in the main page becomes the socket node to the substituted subpage instance and is bound to the port node (place P26 for the deadlock subpage or place P28 for the nondeadlock subpage) in the substituted subpage instance.*

3. *Each port node is a global fusion, i.e. there is a single instance of the given place shared by all sub-page instances in the hierarchical Petri net.*

Figure 2.7 presents the main page of hierarchical Petri net generated for an exemplary computational mesh consisting of 4 elements. Places **Sock[#]** correspond to mesh elements with a given number. All **Sock[#]** places in the main page are input sockets, bound to port nodes of sub-page instances of appropriate type. All **Sock[##]** places in the main page are output sockets, bound to port nodes in sub-pages. Socket nodes in the main page and corresponding port nodes in sub-pages are places by means of which sub-pages are bound to the main page, comprising a coherent model. For precise definitions of socket nodes and port nodes, please refer to [58], page 176. Transitions **SubP12**, **SubP41**, **SubP34** and **SubP23** are substituted with instances of a sub-page modeling a mesh element pair that is adjacent along eighter X or Y axis.

Observation 2.1.3. *Complexity (size of the hierarchical Petri net) of the proposed model can be estimated by the number of adjacent element pairs in a computational mesh (directly determining the number of subpages in the hierarchical Petri net model). This number is highest for rectangular meshes and can be expressed as (#columns * (#rows - 1) + #rows * (#columns - 1)).*

Remark 2.1.4. *The grammar is not deadlock-free.*

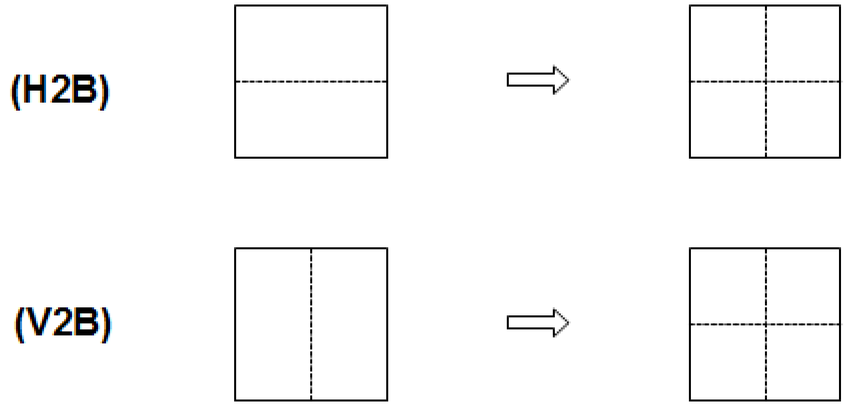


Figure 2.8: New productions in the enhanced graph grammar

Proof. It is clearly visible that the following sequence of fired transitions (in the subpage for a mesh element pair adjacent along X axis): **VBIH1**, **VBIV2**, **BIH1**, **BIV2**, **VVA2**, **VHA1**, **VP2** leads to a dead state. In this state, two mutually contradicting refinement requests have occurred on the right element, leading to a deadlock condition. □

2.1.4. Enhanced grammar

Figure 2.8 presents productions that have been added to the previously defined grammar. Figures 2.9 and 2.10 present the counterparts of the deadlock detecting Petri net subpages reflecting the enhanced grammar, for finite element pairs adjacent along X and Y axis, respectively.

Remark 2.1.5. *The enhanced grammar is deadlock-free.*

Proof. Reachability graph has been generated for given Petri net and given initial marking (shown in the figure). The initial marking reflects the intention of braking each mesh element once. The reachability graph contains no dead state (the Petri net is live) which implies that the grammar modeled by the Petri net is deadlock-free. □

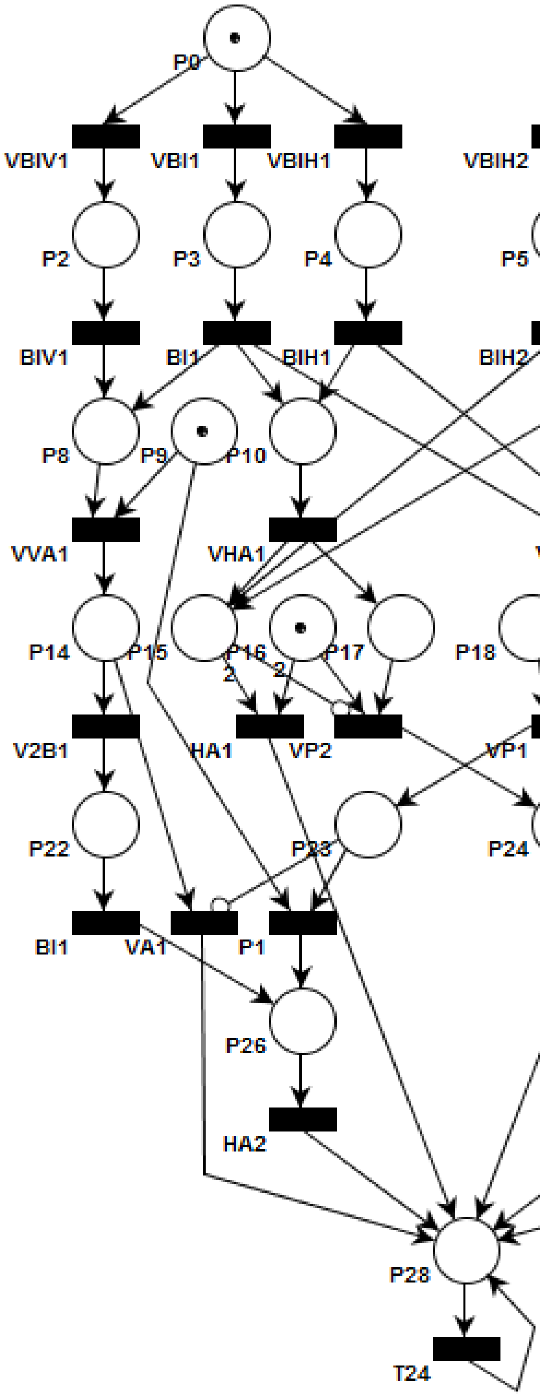


Figure 2.9: Petri net subpage for mesh element pair adjacent along X axis and the enhanced grammar

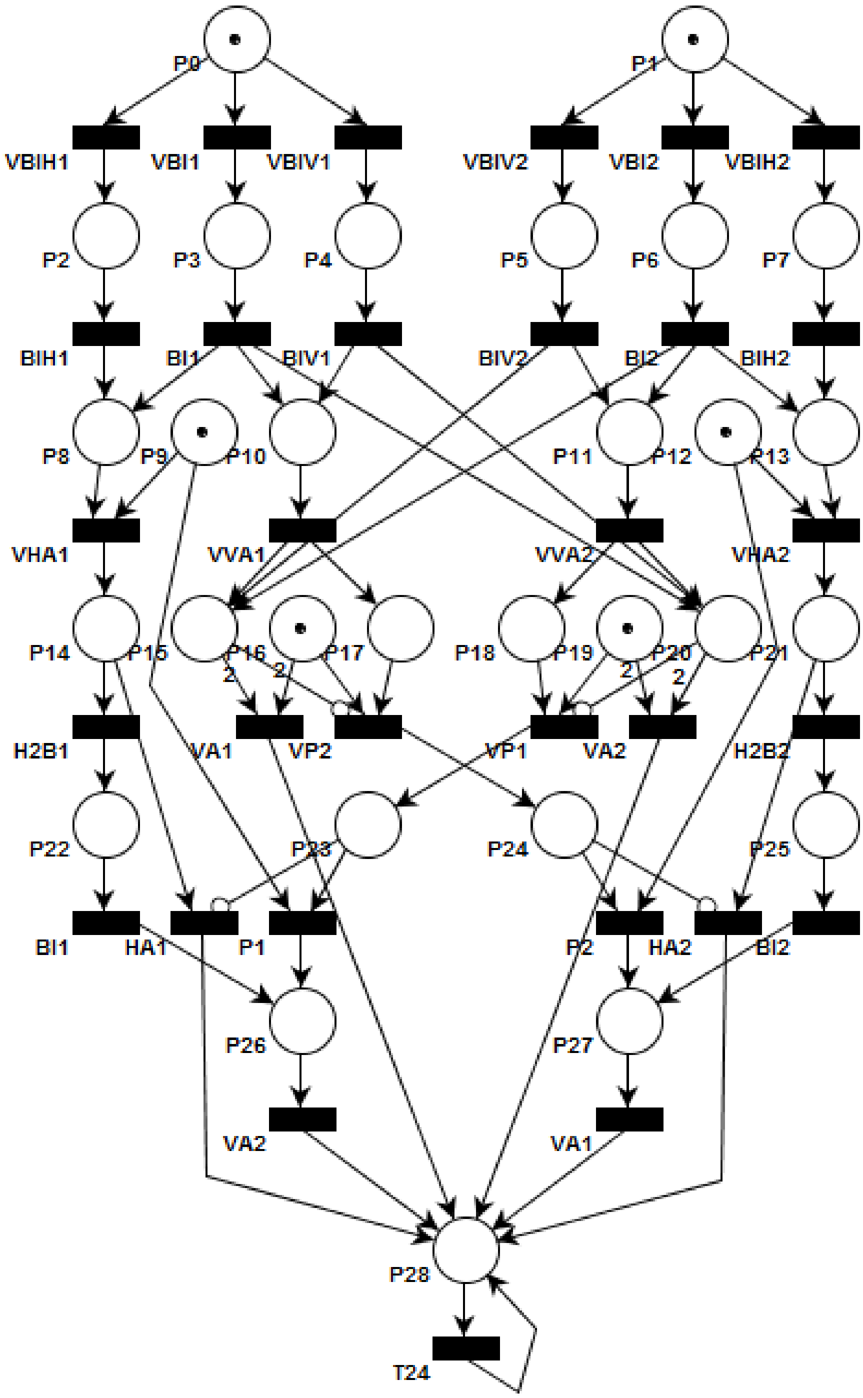


Figure 2.10: Petri net subpage for mesh element pair adjacent along Y axis and the enhanced grammar

2.2. Petri net based detecting of deadlock during anisotropic adaptation of 3D hexahedral meshes

2.2.1. Mesh adaptation algorithm

The mesh adaptation algorithm as implemented in *hp3d* code [11], can be summarized in the following way:

Algorithm 2.2.1. *(Automatic three-dimensinal mesh adaptations)*

```
1   L = List of elements el to be refined with refinement kind
2   do while  L not empty
3     el = get next element from L and its refinement kind
4     loop through face ∈ faces of element el
5       if face belongs to big neighbor element then
6         Store neighbor with its refinement kind at the end of L
7         Store el and its refinement kind at the end of L
8         continue do-loop from line 2
9       endif
10    enddo
11    loop through edge ∈ edges of element el
12      if edge belongs to big neighbor element then
13        Store neighbor with its refinement kind at the end of L
14        Store el and its refinement kind at the end of L
15        continue do-loop from line 2
16      endif
17    enddo
18    break element el in a way kind using the virtual refinement
19  end while
```

Remark 2.2.1. *A* **deadlock** *scenario occurs when a virtual refinement propagates onto some adjacent element either by a face or an edge, and it is contradictory to the virtual refinement that has already been selected for the adjacent element.*

In the mesh adaptation Algorithm 2.2.1, the deadlock happens in line 18 when we try to break the interior of an element that has already been broken in some other way.

2.2.2. Graph grammar model

The mesh transformations performed by the mesh adaptation algorithm can be expressed by several basic tasks, called graph grammar productions. These tasks are summarized in Figure 2.11.

Productions whose names start with **V** denote the so-called virtual refinements - requests for breaking an element interior in one of several possible directions. Productions **(VX)**, **(VY)** and **(VZ)** denote virtual breaking of an element interior in a single direction along X, Y and Z, axis, respectively. Productions **(VXY)**, **(VYZ)** and **(VXZ)** denote virtual breaking of an element interior along two designated axis at the same time. Production **(VXYZ)** denotes virtual breaking of element interior along all three axis at the same time.

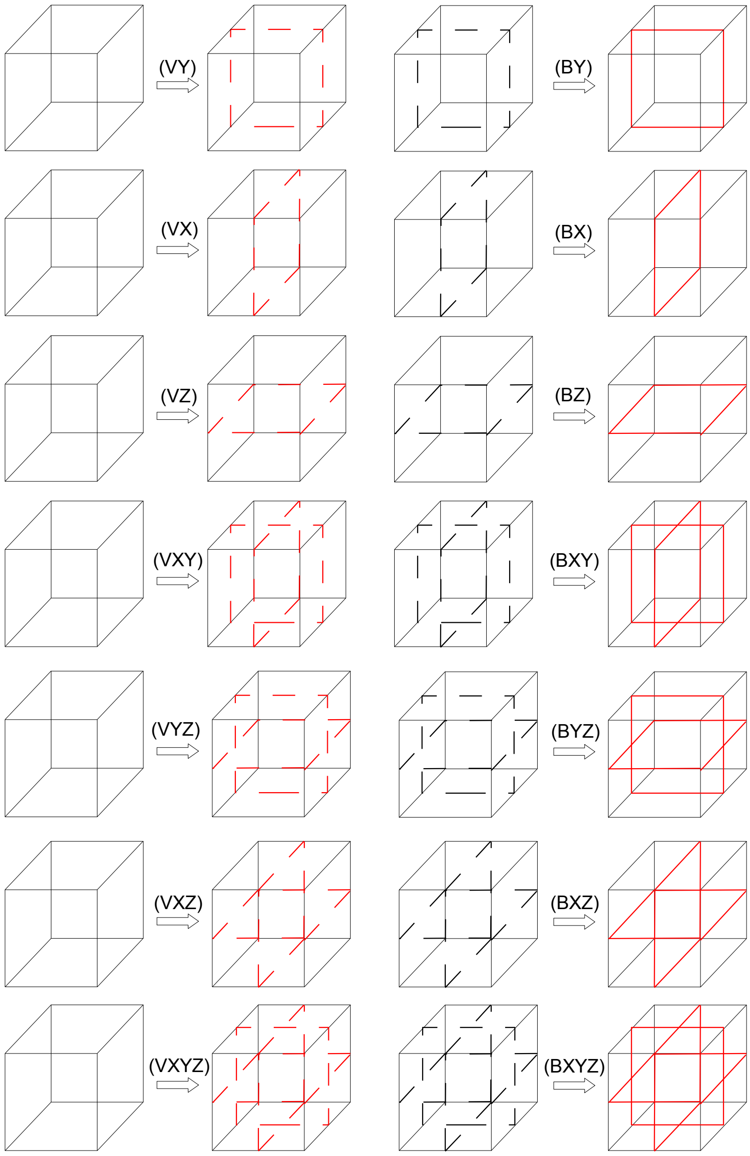


Figure 2.11: Graph grammar productions for the mesh refinements process as implemented in *hp3d* code.

The computational mesh after execution of any of these virtual refinements is not in a legal state. This is because the virtual refinements break only element interiors. Afterwards the mesh must be closed by enforcing additional breaking of some edges. A face must be broken if it is surrounded by two broken interiors. An edge can be broken if it is surrounded by four broken faces. The execution of virtual refinements is followed by the execution of several graph grammar productions, checking the connectivities between edges, faces and interiors, and enforcing some additional refinements. Corresponding graph grammar productions for actual refinements are denoted by **B** letter. Let us assume that closing of the refinement process for faces and edges can be expressed by one production **B*** whose name corresponds to the virtual refinement executed before.

According to 1-irregularity rule in 3D a finite element face can be broken only once. In other words when we try to break a finite element face for the second time, it is necessary to break large adjacent element first. It is assumed that the large adjacent element is always broken in three directions, to prevent long propagation of refinements.

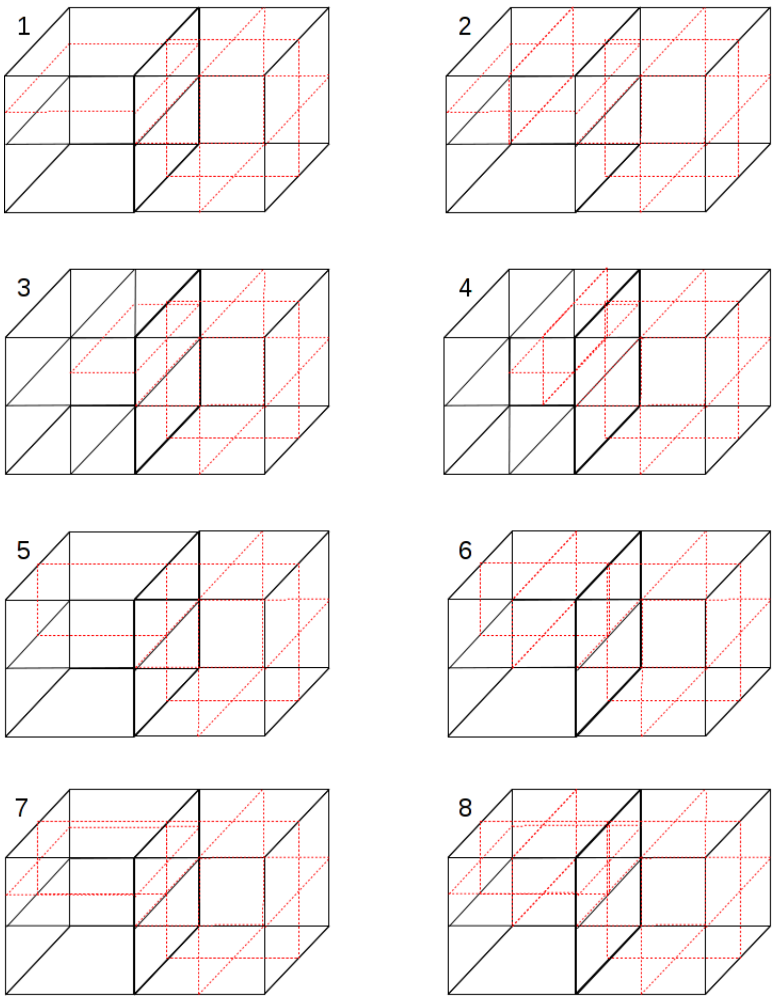


Figure 2.12: Cases 1-8 of propagation of h refinement onto adjacent element.

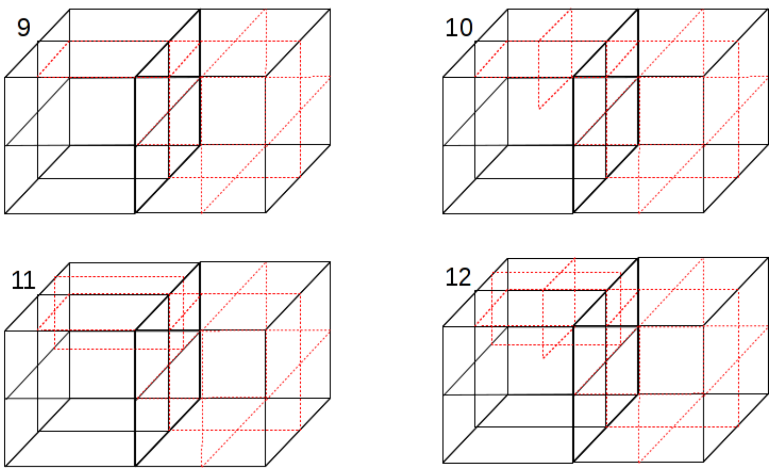


Figure 2.13: Cases 9-12 of propagation of h refinement onto adjacent element.

Observation 2.2.1. *There are fourty eight possible configurations when 1-irregularity rule implies breaking the large adjacent element, as shown in Figures 2.12-2.19.*

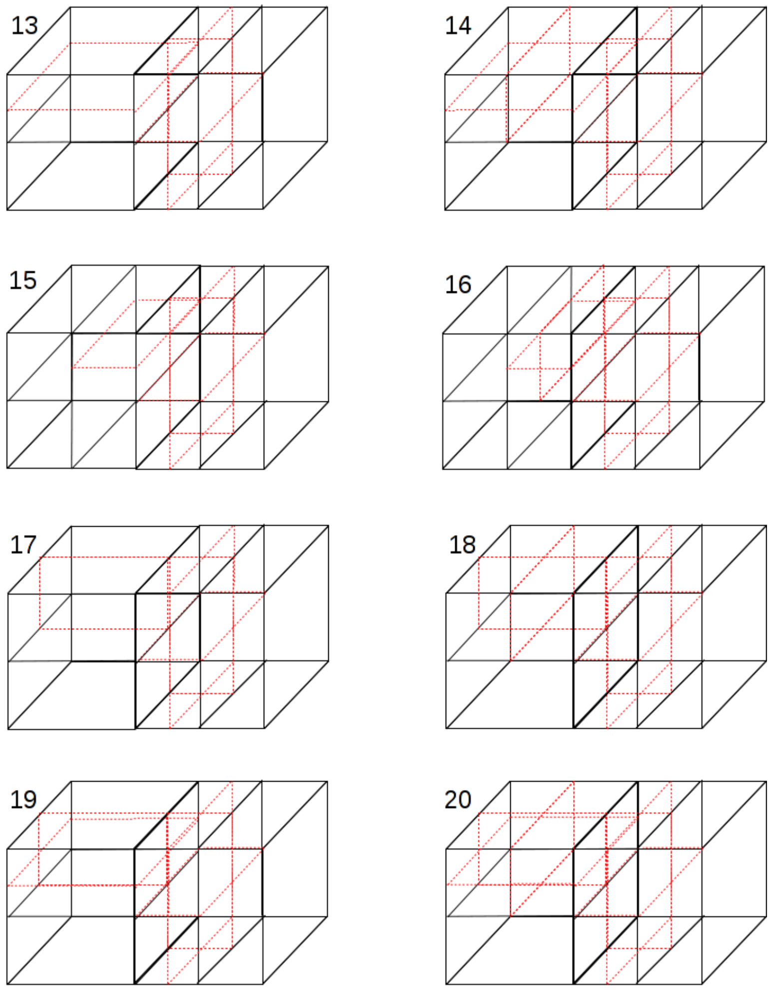


Figure 2.14: Cases 13-20 of propagation of h refinement onto adjacent element.

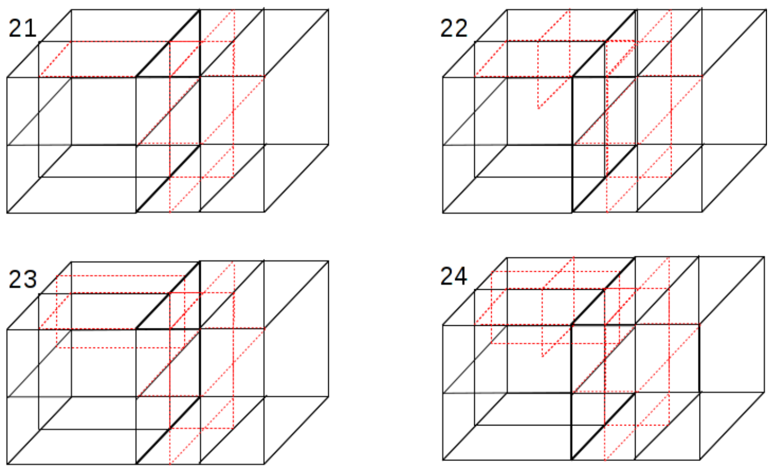


Figure 2.15: Cases 21-24 of propagation of h refinement onto adjacent element.

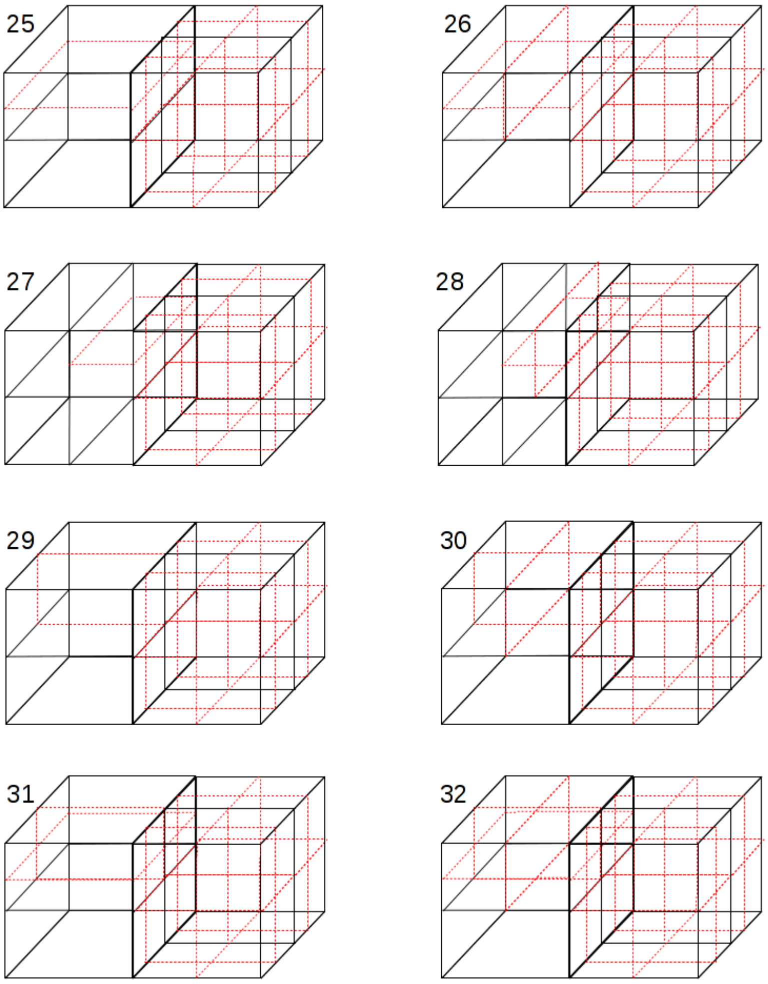


Figure 2.16: Cases 25-32 of propagation of h refinement onto adjacent element.

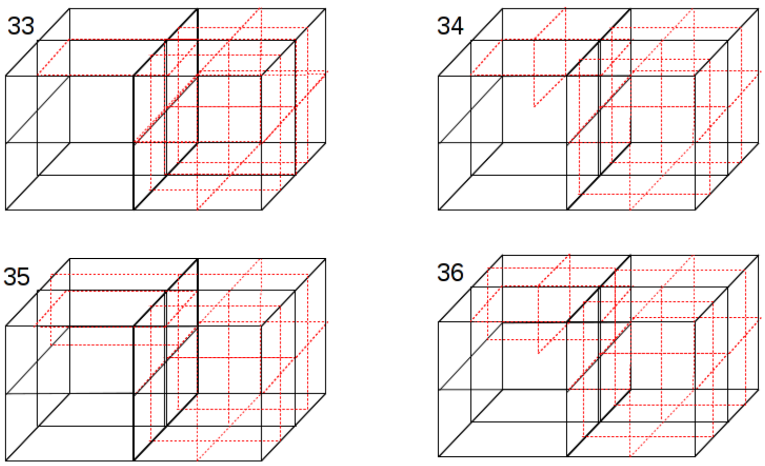


Figure 2.17: Cases 33-36 of propagation of h refinement onto adjacent element.

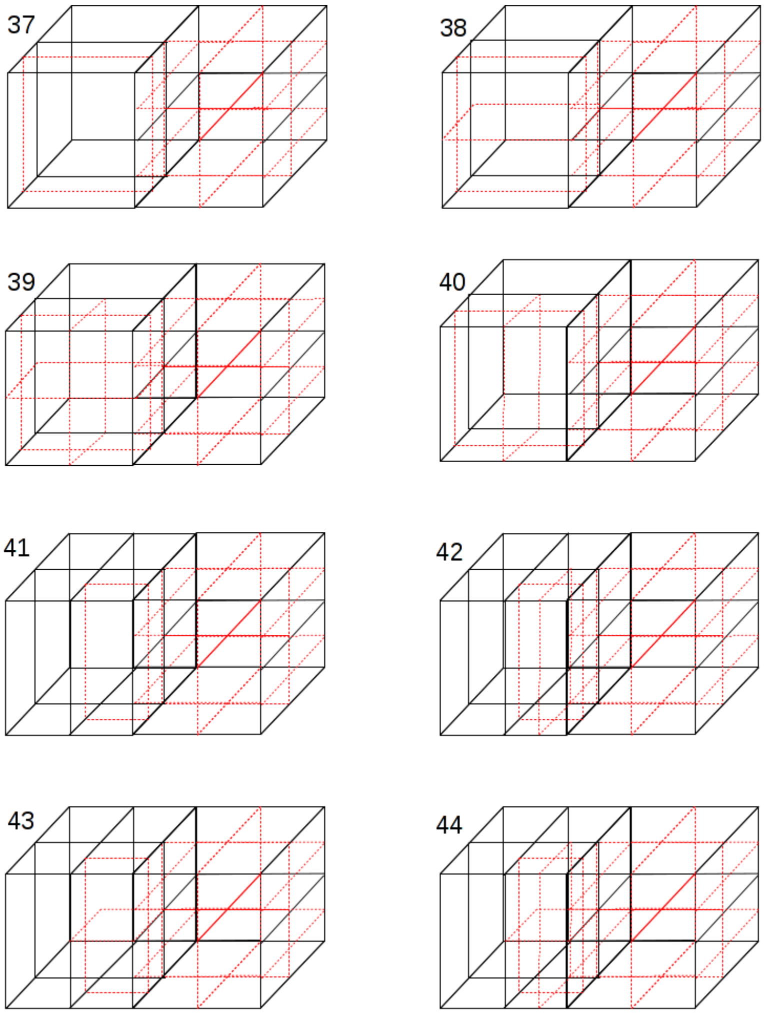


Figure 2.18: Cases 37-44 of propagation of h refinement onto adjacent element.

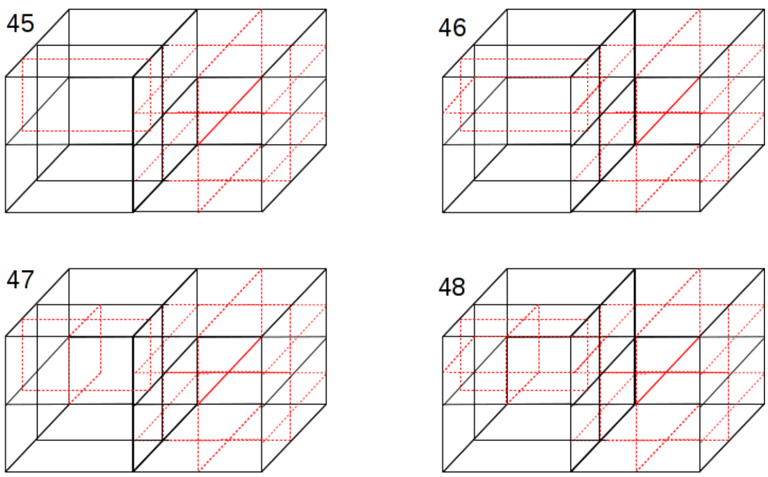


Figure 2.19: Cases 45-48 of propagation of h refinement onto adjacent element.

2.2.3. Hierarchical Petri net model

Remark 2.2.2. *In order to detect a possible deadlock scenario, we need to construct a hierarchical Petri net with the main page covering the entire mesh, and sub-pages corresponding to pairs of elements either through a face or through an edge.*

The hierarchical Petri net model is constructed for the entire mesh, with main page corresponding to the entire mesh, and with sub-pages corresponding to all element pairs, adjacent either by face or by edge. Because of the 1-irregularity rule enforced over the entire mesh, all the pairs of elements are at the same level of adaptation. Each sub-page corresponding to a single pair of elements considers refinement request for any of the elements in the pair, with possible propagation to the other element from the pair. The sub-page considers also all possible refinement requests coming from the external elements. Thus, let us consider all possible combinations of two virtual refinement requests, and check if they can result in a deadlock.

The hierarchical Petri net model has been constructed in such a way that actual deadlock detection is performed by the sub-pages covering two-element patches of the mesh. The hierarchical Petri net sub-page for finite elements adjacent along X axis is depicted in Figures 2.20 and 2.21. Similar hierarchical Petri net sub-pages for finite elements adjacent along Y and Z axis are presented in Figures 2.22 and 2.23, respectively.

The Petri nets described in this section are again defined as hierarchical colored Petri nets (compare [58] p. 177 definition 10.16), limited to just one color (there is just one type of token). Like for the two-dimensional case, sub-pages are linked to the main page by socket and port nodes. A socket (in the main page) and its corresponding port (in a sub-page) behave as a single common place (or a fusion) shared between the two pages (compare [58] p. 176).

Since the propagation of refinements may also occur by an edge, as it is depicted in Figure 1.8, it is also necessary to consider patches of elements adjacent through edges. Each element has up to six neighbors through faces, where there are actually three symmetric Petri nets, for adjacency along X, Y and Z axis. There are also twelve edges, and there are twelve possible adjacent neighbors through edges. The Petri net sub-page for adjacency by edges is similar to the sub-pages reflecting adjacency by faces, but only the refinements in the direction perpendicular to the edge may occur there. Such a Petri net is presented in Figures 2.24 and 2.25.

The Petri net arcs define all possible execution paths for a round of mesh element refinements by enforcing dependency relationships between relevant transitions (productions). Whenever only one of a set of grammar productions can be executed, the corresponding Petri net transitions depend on a common place with a single token in the initial marking. Whenever execution of a production blocks execution of another production, an inhibitor arc is used between the corresponding Petri net transitions (actually between the intermediate place and the dependent transition).

Each hierarchical Petri net subpage contains two starting places (**P1** and **P2**) - one for each mesh element of the modeled pair. **P1** and **P2** are fusion places - shared between all sub-pages covering common mesh elements (a given mesh element can be part of up to six element pairs). The two upper rows of Petri net transitions are named after the grammar productions they represent. Numbers at the end of transition names denote the corresponding mesh element to which a given transition pertains. Firing any of those transitions models executing a corresponding grammar production. The remaining Petri net transitions model the following mesh element transformations:

- **VXA** - request (virtual) to break along X axis the sub-element adjacent to the other element in the pair
- **VYA** - request (virtual) to break along Y axis the sub-element adjacent to the other element in the pair
- **VZA** - request (virtual) to break along Z axis the sub-element adjacent to the other element in the pair
- **VXYA** - request (virtual) to break along X and Y axis the sub-element adjacent to the other element in the pair

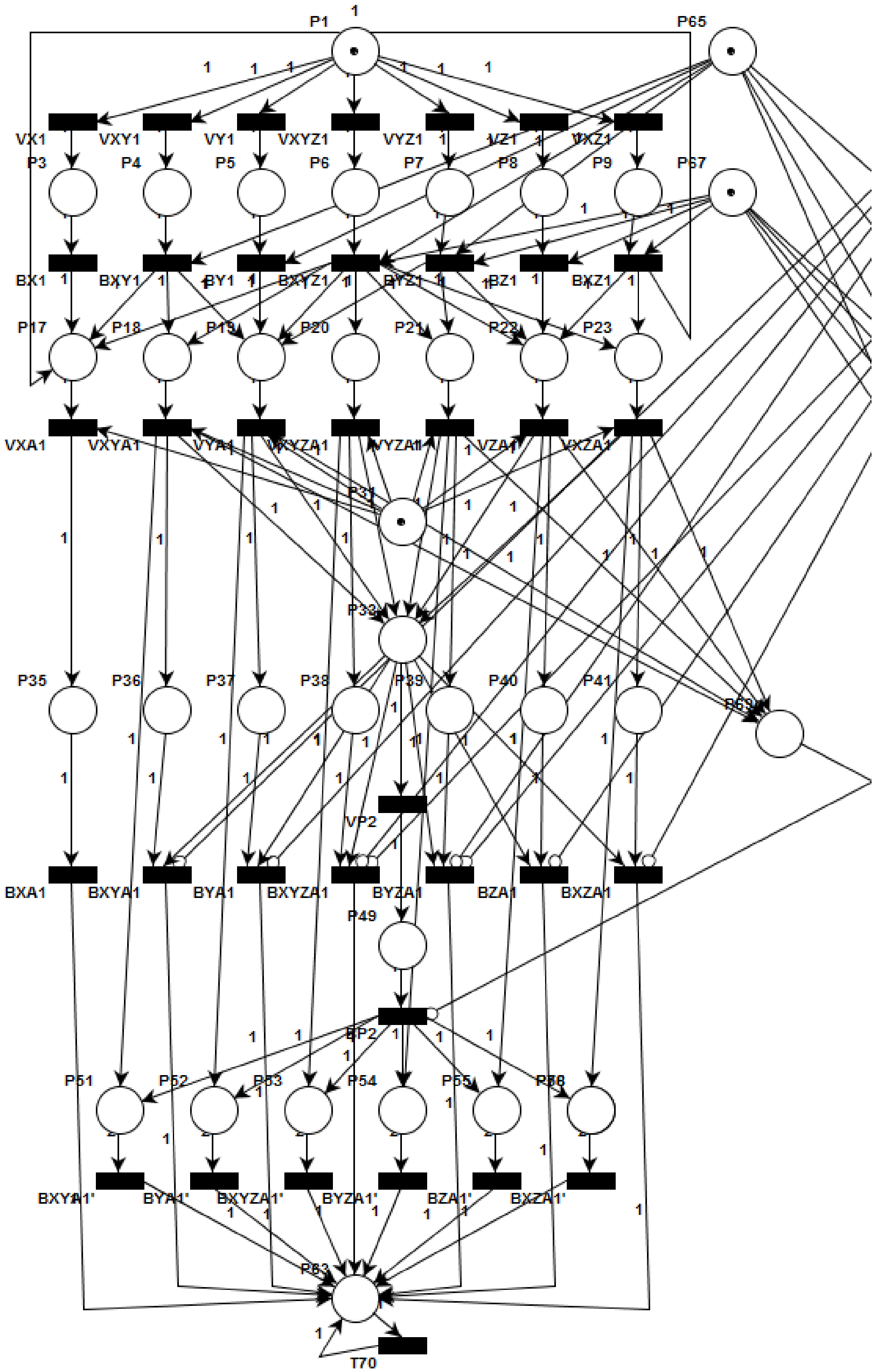


Figure 2.20: First part of the hierarchical Petri net subpage with deadlock for finite elements adjacent along X axis.

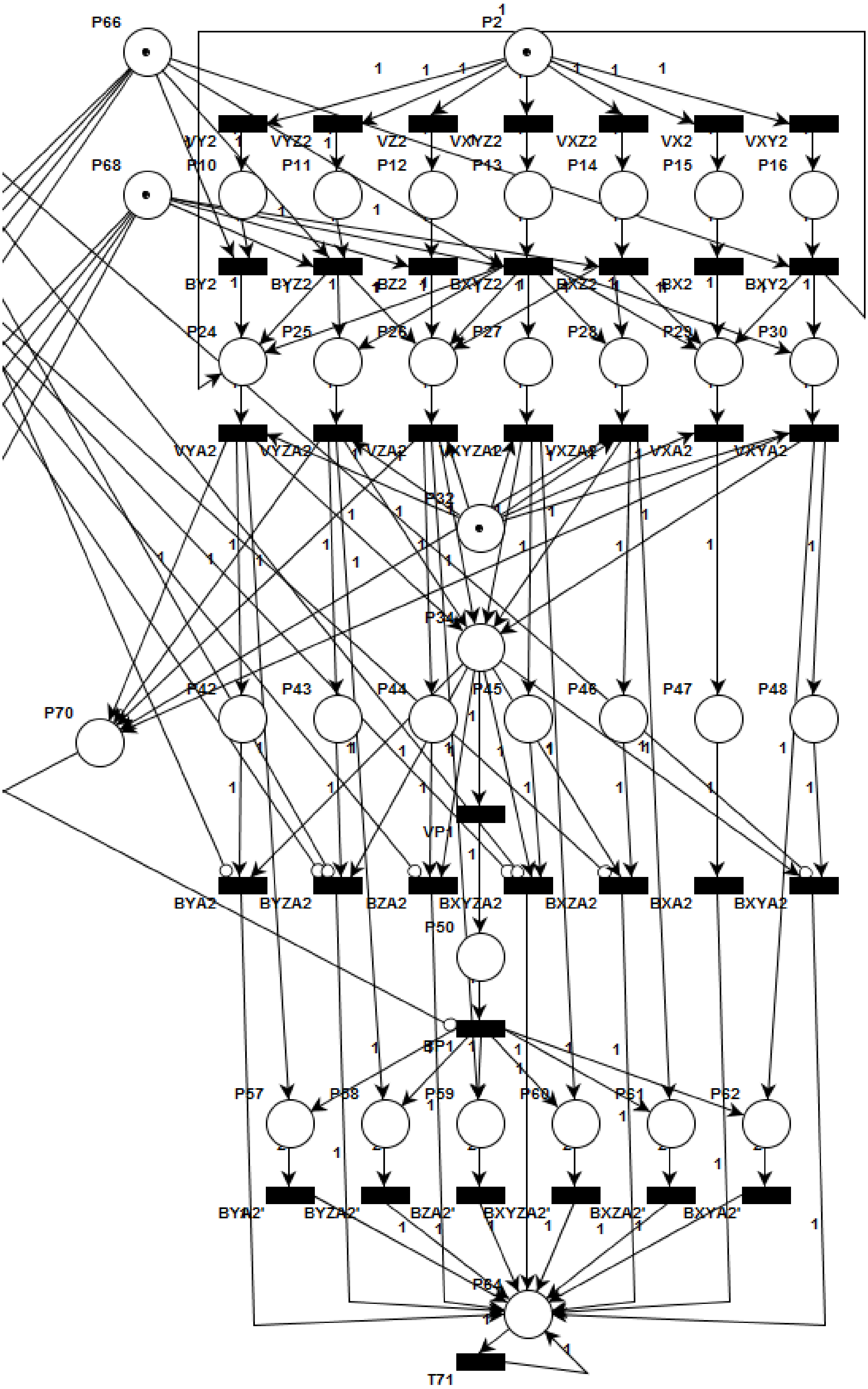


Figure 2.21: Second part of the hierarchical Petri net subpage with deadlock for finite elements adjacent along X axis.

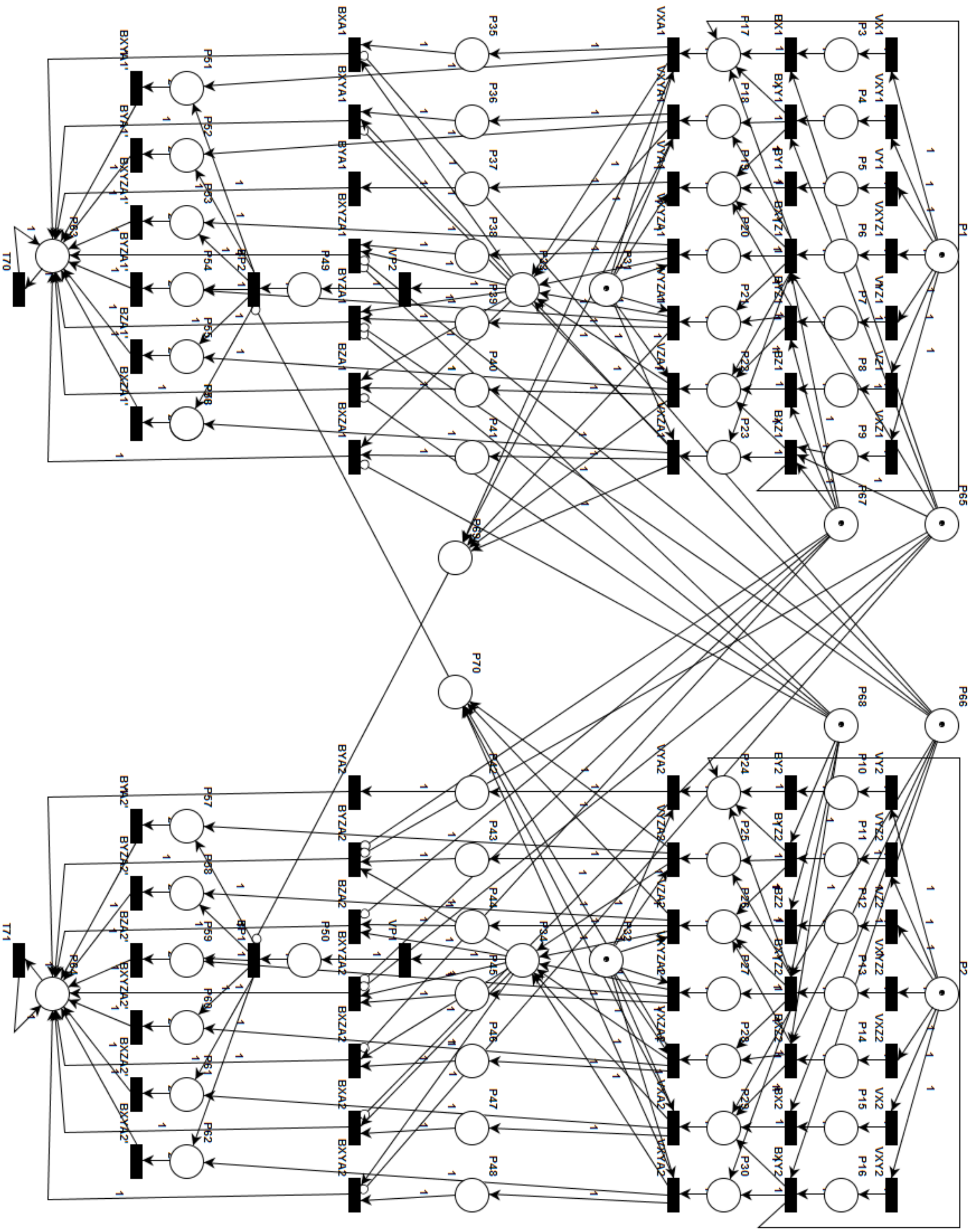


Figure 2.22: The hierarchical Petri net subpage with deadlock for finite elements adjacent along Y axis.

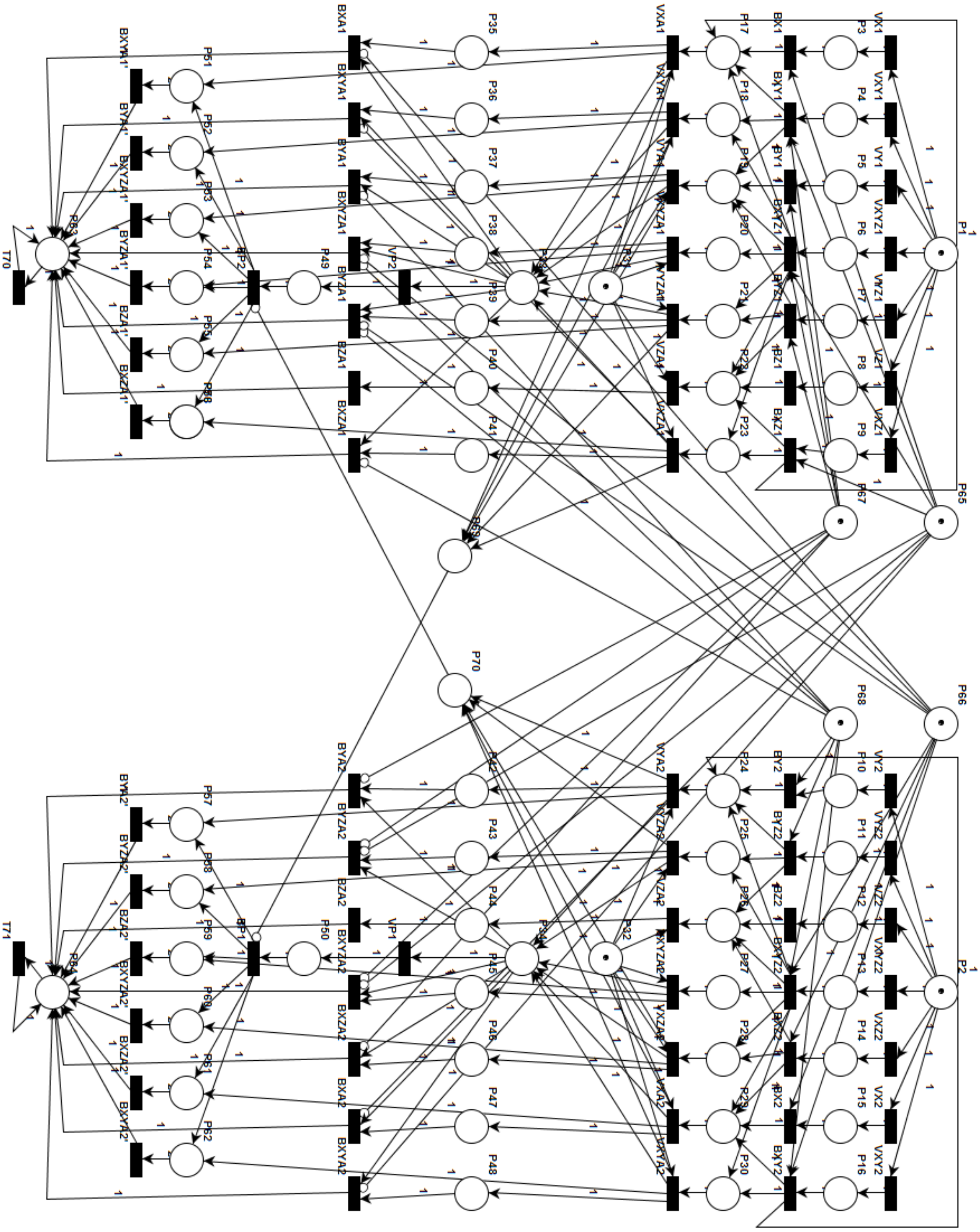


Figure 2.23: The hierarchical Petri net subpage with deadlock for finite elements adjacent along Z axis.

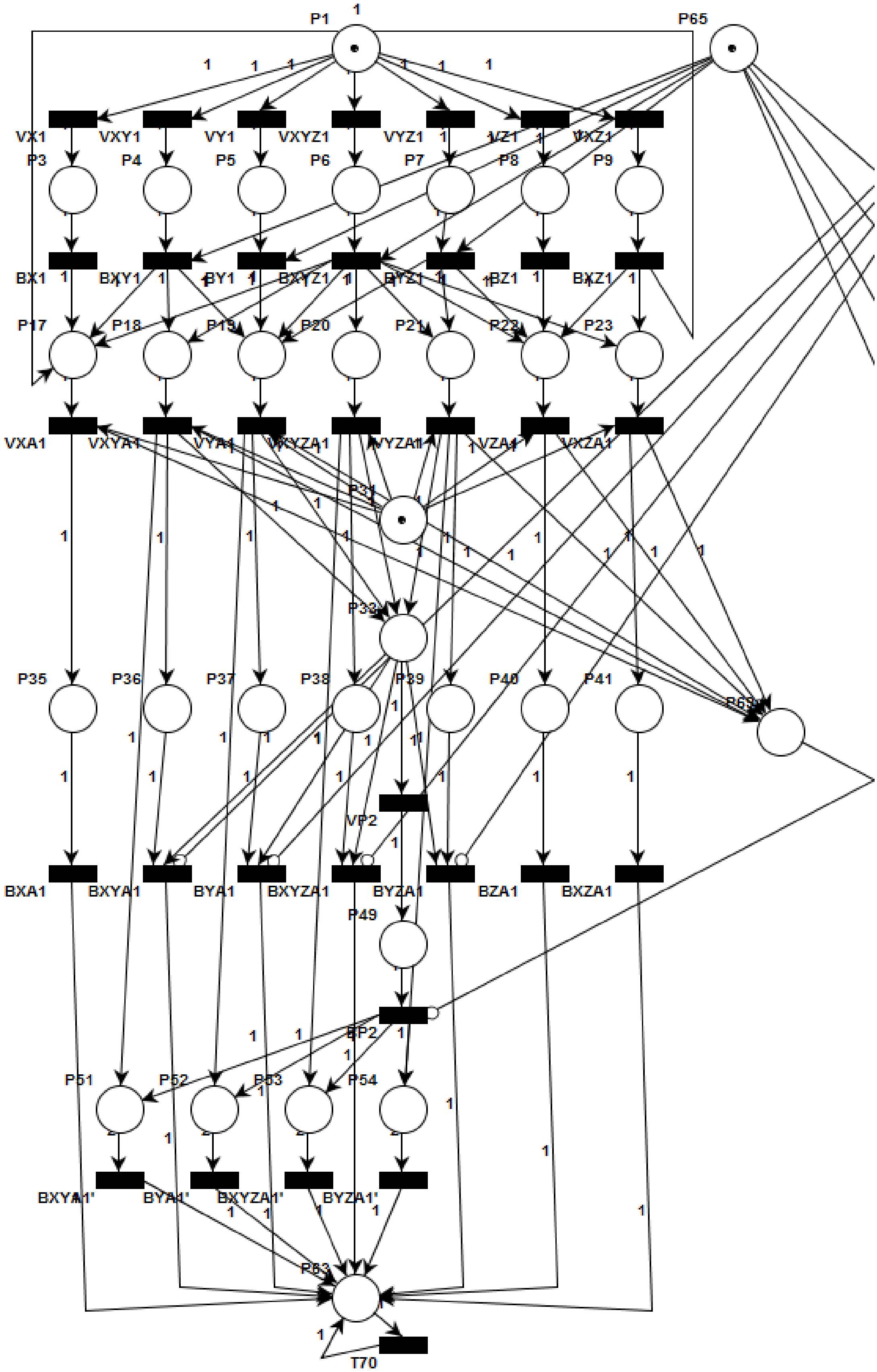


Figure 2.24: First part of the hierarchical Petri net subpage with deadlock for finite elements adjacent by edge.

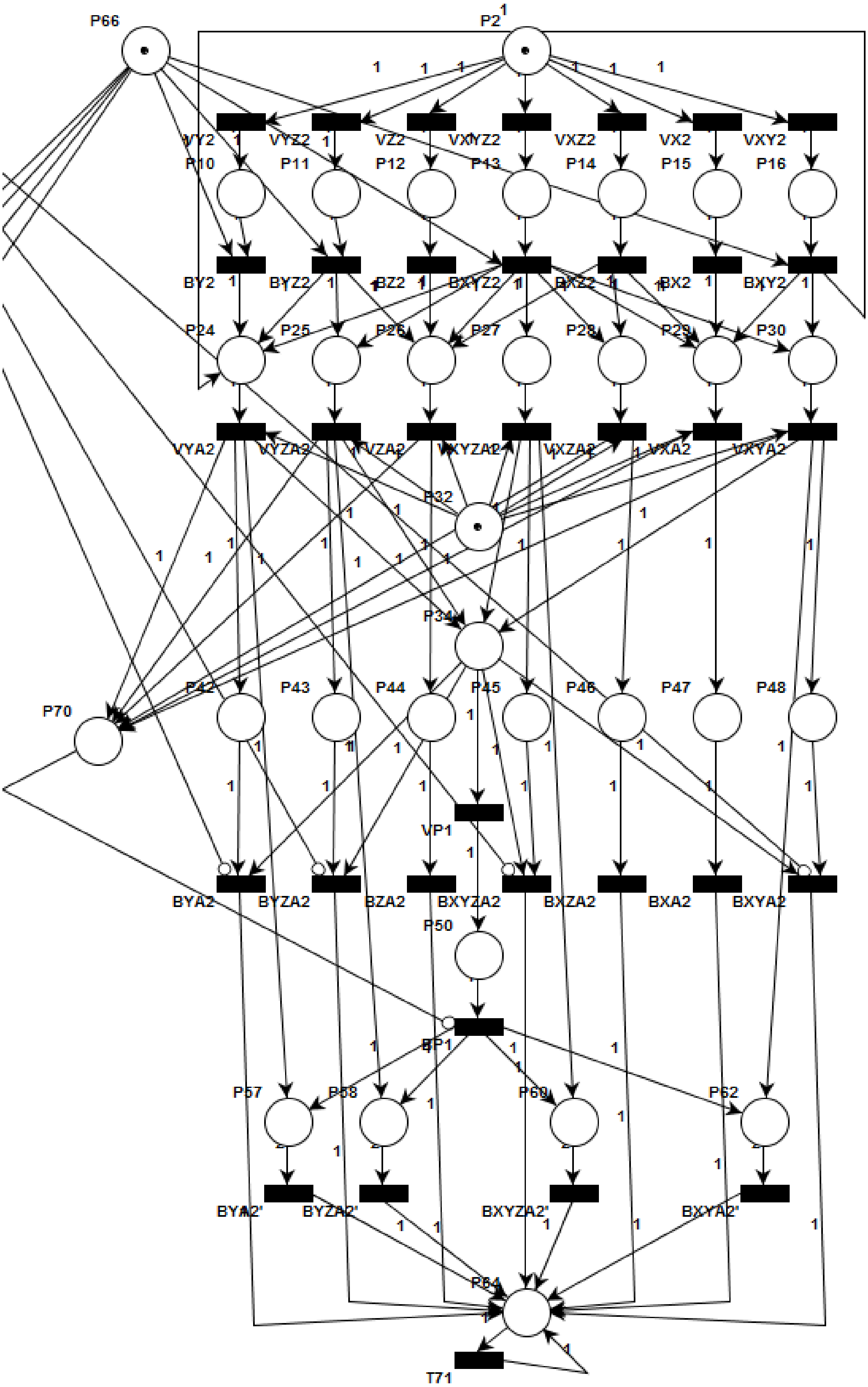


Figure 2.25: Second part of the hierarchical Petri net subpage with deadlock for finite elements adjacent by edge.

- **VXZA** - request (virtual) to break along X and Z axis the sub-element adjacent to the other element in the pair
- **VYZA** - request (virtual) to break along Y and Z axis the sub-element adjacent to the other element in the pair
- **BXA** - transformation breaking along X axis the sub-element adjacent to the other element in the pair
- **BYA** - transformation breaking along Y axis the sub-element adjacent to the other element in the pair
- **BZA** - transformation breaking along Z axis the sub-element adjacent to the other element in the pair
- **BXYA** - transformation breaking along X and Y axis the sub-element adjacent to the other element in the pair
- **BXZA** - transformation breaking along X and Z axis the sub-element adjacent to the other element in the pair
- **BYZA** - transformation breaking along Y and Z axis the sub-element adjacent to the other element in the pair
- **BXYZA** - transformation breaking along all 3 axis the sub-element adjacent to the other element in the pair
- **VP** - transformation propagating the refinement request (virtual) onto the other element in the pair
- **BP** - transformation executing the propagated refinement
- **VB** - transformation converting any virtual refinement into a three-directional virtual refinement

Transitions whose names end with prim model the same graph transformations as the corresponding transitions without prim at the end of the name but reachable by a different execution path (that is, with vs. without refinement propagation).

A mesh (sub-)element can be broken for the second time (transitions **B [D]A[#]**, where **[D]** stands for any direction(s) and **[#]** is the number of concerned mesh element) only when the adjacent element is already broken in the same direction at least once. This can be achieved in either of the following two ways:

1. The adjacent element has been broken in the required direction independently.
2. The required refinement is propagated onto the adjacent element (e.g. **P33** $\rightarrow$ **VP2** $\rightarrow$ **P49** $\rightarrow$ **BP2** for the "left" element in the pair).

An alternative to the above two scenarios is modeled in the Petri net by means of places **P65 - P70**. Single breaking of a mesh element in a set **D** of directions "un-inhibits" (unblocks) the single breaking of the adjacent (sub-)element in a subset **D** of directions (places **P65 - P68**). A second virtual refinement of a mesh element in fewer than three directions at the same time inhibits the refinement to be propagated from the adjacent mesh element (places **P69** and **P70**). It is critical that each pair of adjacent mesh elements is covered with a Petri net subpage of appropriate type. To this end, the hierarchical Petri net generation algorithm for a given finite element mesh has been developed.

Assumption 2.2.1. *All elements of the mesh to be analyzed are at the same adaptation level. This is a direct consequence of the 1-irregularity rule that must be fulfilled over the mesh.*

Algorithm 2.2.2. *Generation of a hierarchical Petri net for a given 3D finite element mesh*

1. *Create the main page of the hierarchical Petri net.*
   - *Create a Petri net place for each element of the mesh being analyzed.*
   - *For each pair of adjacent (by face in any direction: along X, Y or Z axis; or by any of the twelve edges) mesh elements, create a Petri net transition and connect the corresponding places to this transition with arcs.*

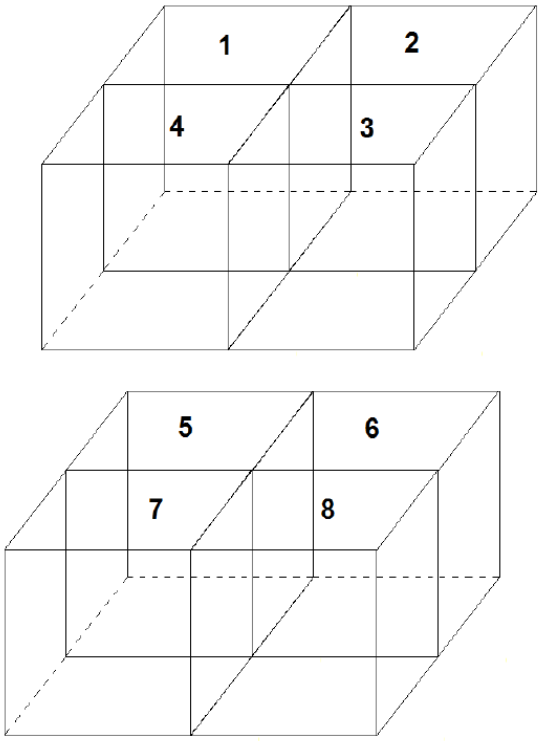


Figure 2.26: Exemplary eight finite element mesh.

  - *Create two output places for each transition and connect each place to its corresponding transition with an arc.*

2. *Bind the main page of the hierarchical Petri net with the sub-pages.*
   - *Substitute each transition in the main page with an instance of appropriate sub-page type, depending on whether the input places to the given transition represent mesh elements that are face-adjacent along X, Y or Z axis, or adjacent by one of the twelve edges.*
   - *The input places of each transition in the main page become the socket nodes to the substituted sub-page instance and are bound to the port nodes (places* **P1** *and* **P2***) in the substituted sub-page instance.*
   - *The output places of each transition in the main page become the socket nodes to the substituted sub-page instance and are bound to the port nodes (places* **P63** *and* **P64***) in the substituted sub-page instance.*
3. *Each port node is a global fusion, i.e. there is a single instance of given place shared by all sub-page instances of the hierarchical Petri net.*

Figure 2.27 presents the main page of the hierarchical Petri net generated for an exemplary computational mesh consisting of 8 elements, depicted in Figure 2.26. Places **Elem[#]** correspond to mesh elements with given number. All **Elem[#]** places in the main page are input sockets, bound to port nodes (places **P1** and **P2**) of sub-page instances of appropriate type. All **P[#]** places in the main page are output sockets, bound to port nodes (places **P63** and **P64**) in sub-pages. Socket nodes in the main page and corresponding port nodes in sub-pages are places by means of which sub-pages are bound to the main page, comprising a coherent model. For precise definitions of socket nodes and port nodes, please refer to [58], page 176. Transitions **Face12**, **Face34**, **Face56** and **Face78** are substituted with instances of a sub-page modeling a mesh element pair that is face-adjacent along X axis. Transitions **Face14**,

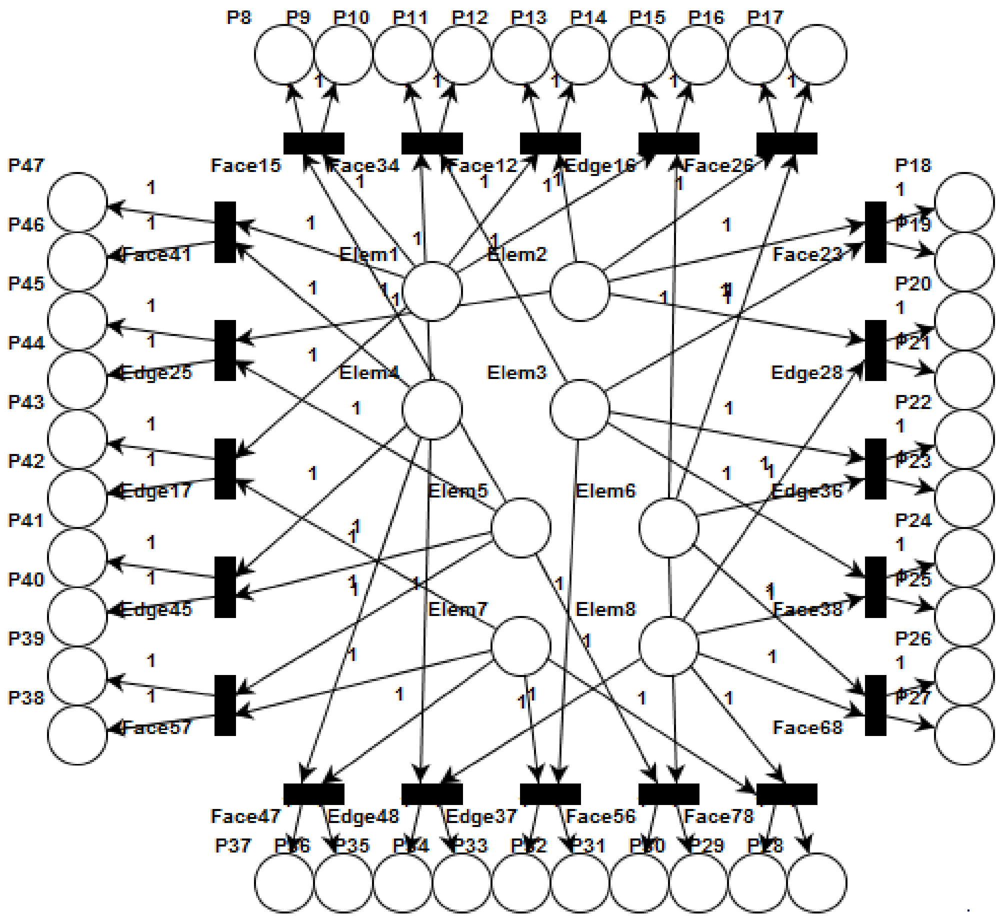


Figure 2.27: The main page of the hierarchical Petri net for the eight element mesh example.

Face23, **Face57** and **Face68** are substituted with instances of a sub-page modeling a mesh element pair that is face-adjacent along Y axis. Transitions **Face15**, **Face26**, **Face38** and **Face47** are substituted with instances of a sub-page modeling a mesh element pair that is face-adjacent along Z axis. Transitions **Edge[#][#]** are substituted with instances of a sub-page modeling a mesh element pair that is edge-adjacent in appropriate direction.

Remark 2.2.3. *Complexity (size of the hierarchical Petri net) S of the proposed model can be estimated by the number of adjacent element pairs in a computational mesh (directly determining the number of sub-pages in the hierarchical Petri net model). This number is highest for (regular) hexahedral meshes and can be expressed as*

$$S = F + E$$
$$F = ((x-1)yz + x(y-1)z + xy(z-1))$$
$$E = 2(((min(x,y)-1)max(x,y) + min(x,y)+1)z + ((min(y,z)-1)max(y,z) + min(y,z)+1)x) \tag{2.1}$$

where:
F - the number of face-adjacent element pairs;
E - the number of edge-adjacent element pairs;
and x, y, z denote the number of elements in X, Y and Z directions, respectively.

As far as the number of reachable Petri net's states is concerned, I used PIPE software to compute them automatically for each sub-page type. I obtained the following numbers:

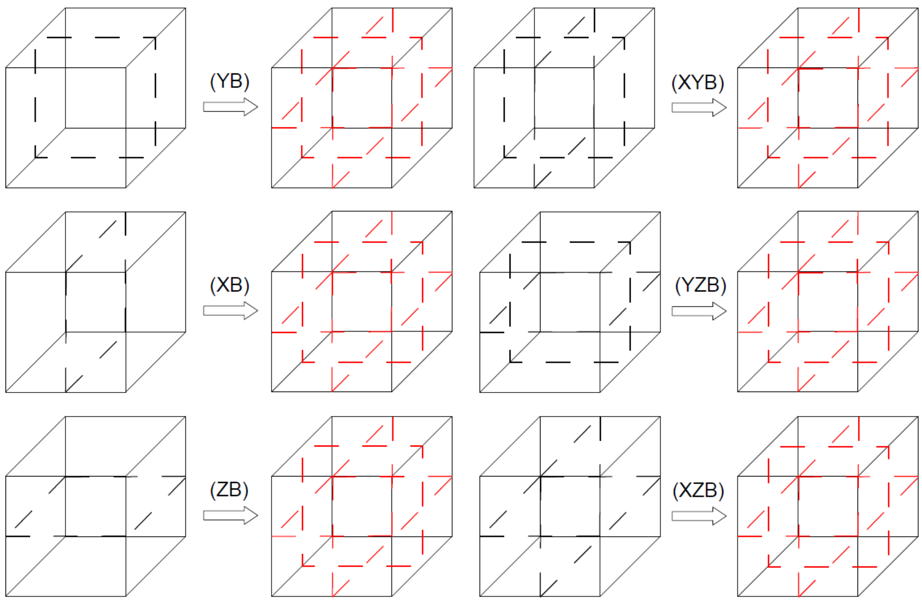


Figure 2.28: Additional graph grammar productions to eliminate the deadlock problem.

- 5391 states for face-adjacent element pair in deadlock prone graph grammar,
- 3918 states for edge-adjacent element pair in deadlock prone graph grammar,
- 5859 states for face-adjacent element pair in deadlock free graph grammar, and
- 3918 states for edge-adjacent element pair in deadlock free graph grammar.

Remark 2.2.4. *The grammar is not deadlock-free.*

Proof. It is clearly visible that the following sequence of fired transitions (in the subpage for a mesh element pair adjacent along X axis) **VY1**, **BY1**, **VZ2**, **BZ2**, **VYA1**, **VZA2**, **VP2**, **VP1** leads to a dead state. In this state, two mutually contradicting refinement requests have occurred on both pair elements, leading to a deadlock scenario.

2.2.4. Enhanced grammar

In this section, some additional graph grammar productions are provided, which allow to overcome the deadlock problem. Figure 2.28 presents productions that have been added to the previously defined grammar. These graph grammar productions update the broken interior of an element in order to merge two different refinement requests. The implementation of these additional graph grammar productions in the mesh adaptation Algorithm 3Dadaptation requires replacing line 18 with the following lines:

```
18a   if element el interior is already broken then
18b     replace the virtual refinement of element el with the mixture of
18c     actual_refinement_kind and the new refinement kind
18d   else
18e     break element el in a way kind using the virtual refinement
18f   endif
```

Figures 2.29 and 2.30 present the counterparts of the deadlock detecting Petri net sub-page reflecting the enhanced grammar, for finite element pairs adjacent along X axis. It is also possible to construct analogous Petri nets for elements adjacent along Y and Z axis, as presented in Figures 2.31 and 2.32 . The corresponding deadlock-free Petri net sub-page for elements adjacent through an edge is also presented in Figures 2.33 and 2.34. Transitions **VB1** and **VB2** have been added to the hierarchical Petri net sub-pages, with arcs from places **P69** and **P70**, respectively.

Firing the newly added transitions effectively "un-inhibits" (unblocks) transitions **BP2** and **BP1** respectively, should any of the latter had been previously inhibited (by firing a transition representing a contradicting refinement request). This result demonstrates that executing any of the newly added grammar productions reconciles the contradicting refinement requests. Additionally, arcs **BP2** $\rightarrow$ **P64** and **BP1** $\rightarrow$ **P63** have been added to reflect the fact that actual execution of the propagated refinement brings a given mesh element to the next adaptation level.

Remark 2.2.5. *The enhanced grammar is deadlock-free.*

Proof. Reachability graph has been generated from PIPE [9] for a given Petri net and given initial marking (shown in the figures). The initial marking reflects the intention of breaking each mesh element once. The reachability graph contains no dead state (the Petri net is alive), which implies that the grammar modeled by the Petri net is deadlock-free.

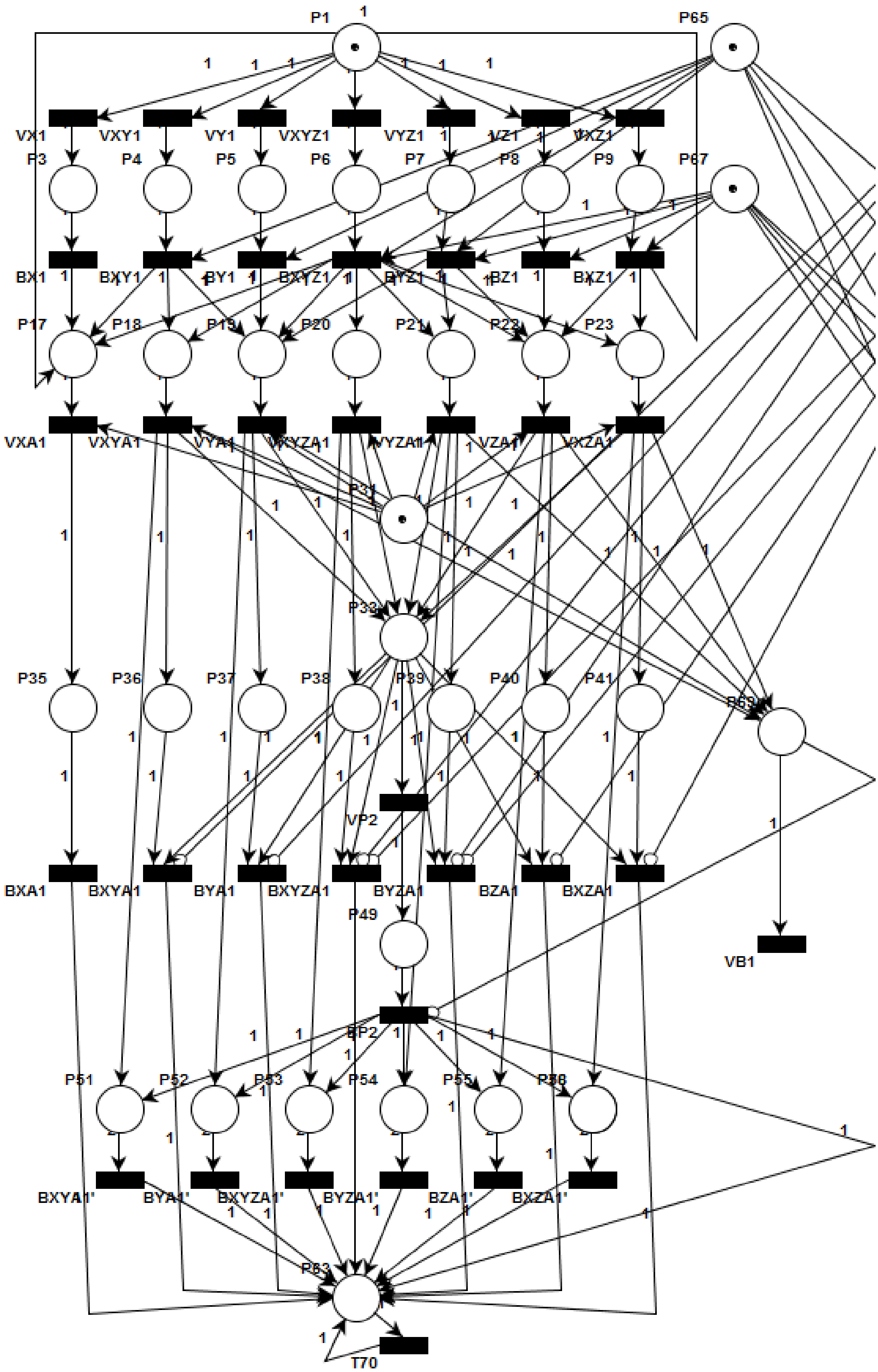


Figure 2.29: First part of deadlock-free hierarchical Petri net subpage for finite elements adjacent along X axis.

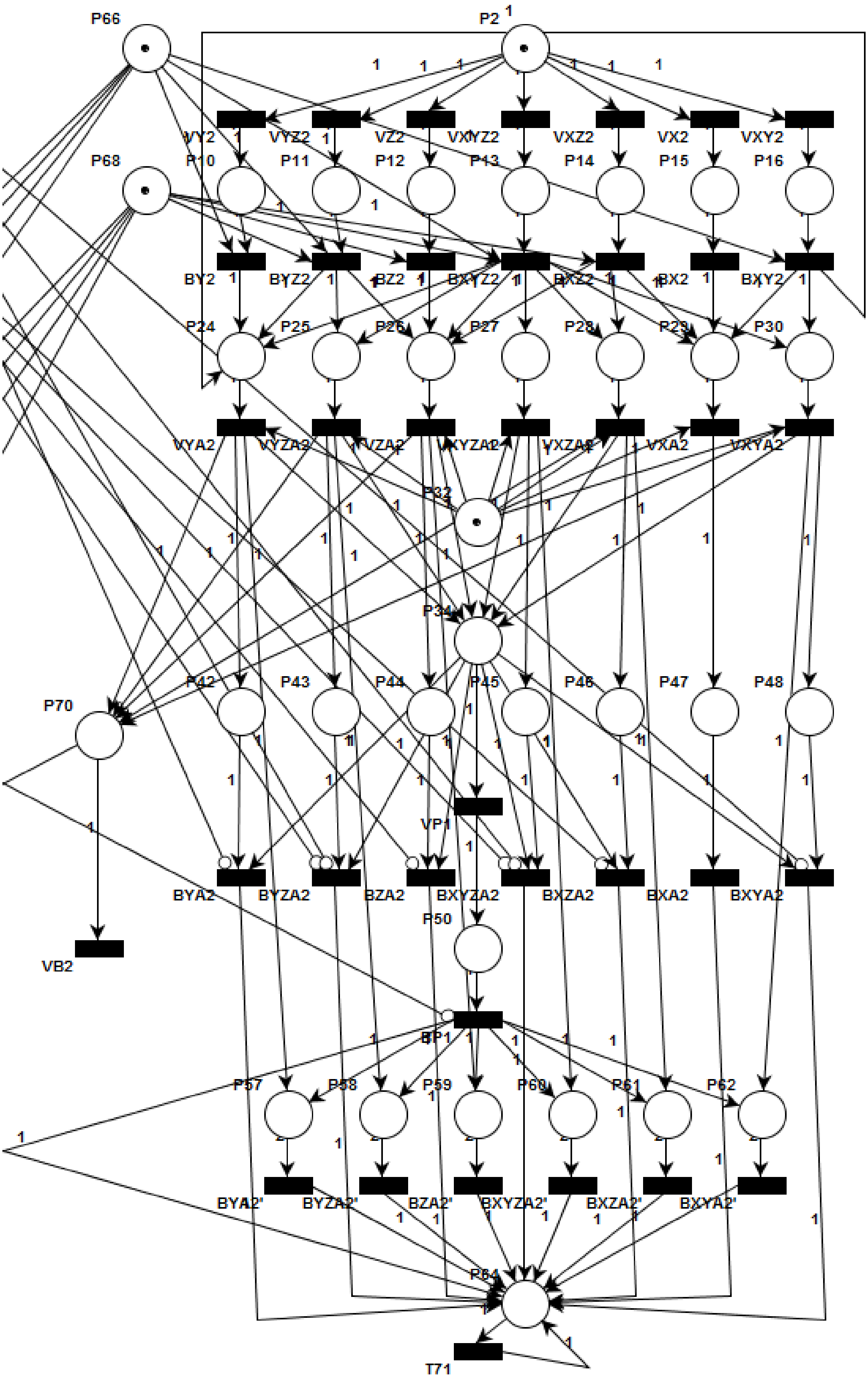


Figure 2.30: Second part of deadlock-free hierarchical Petri net subpage for finite elements adjacent along X axis.

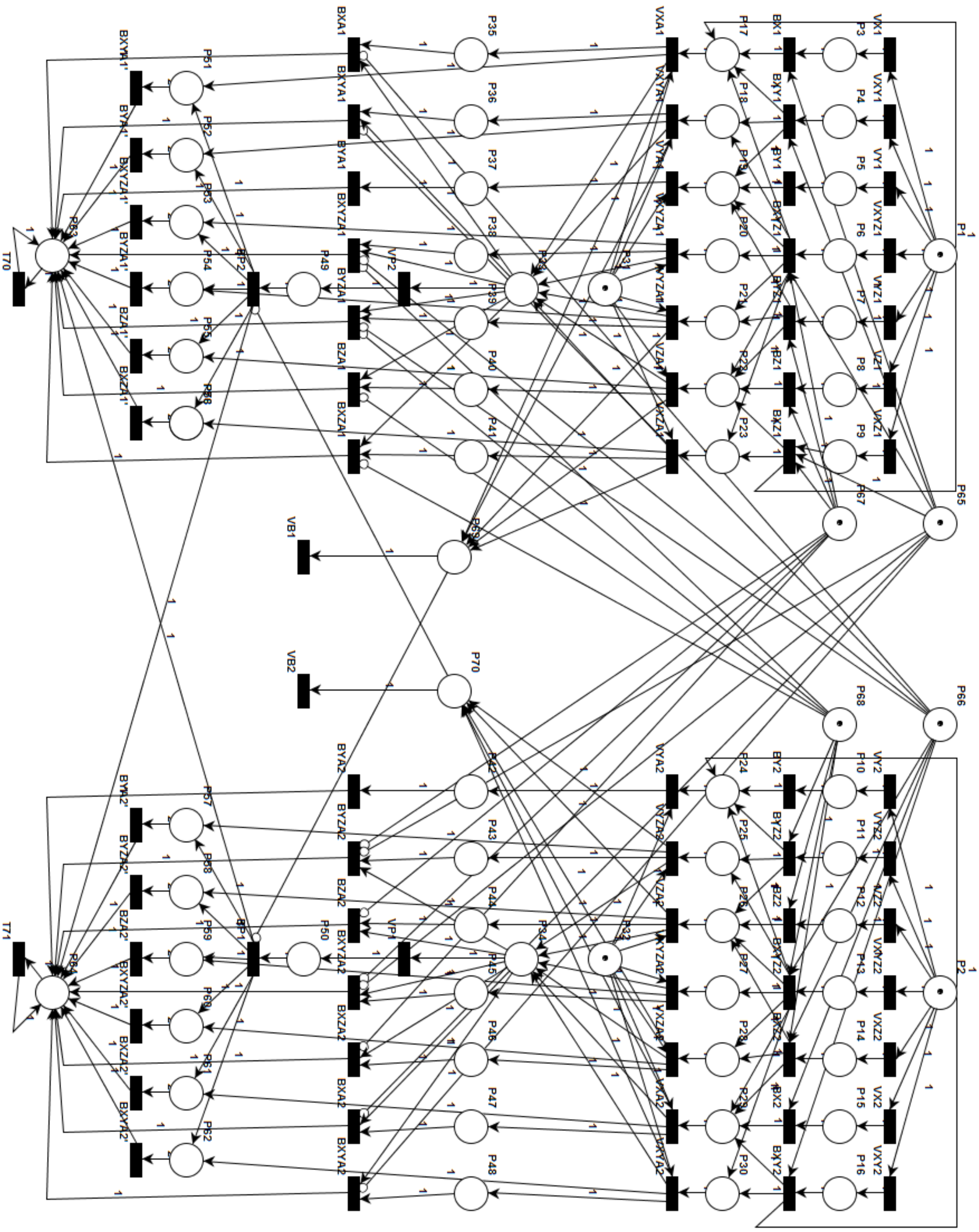


Figure 2.31: Deadlock-free hierarchical Petri net subpage for finite elements adjacent along Y axis.

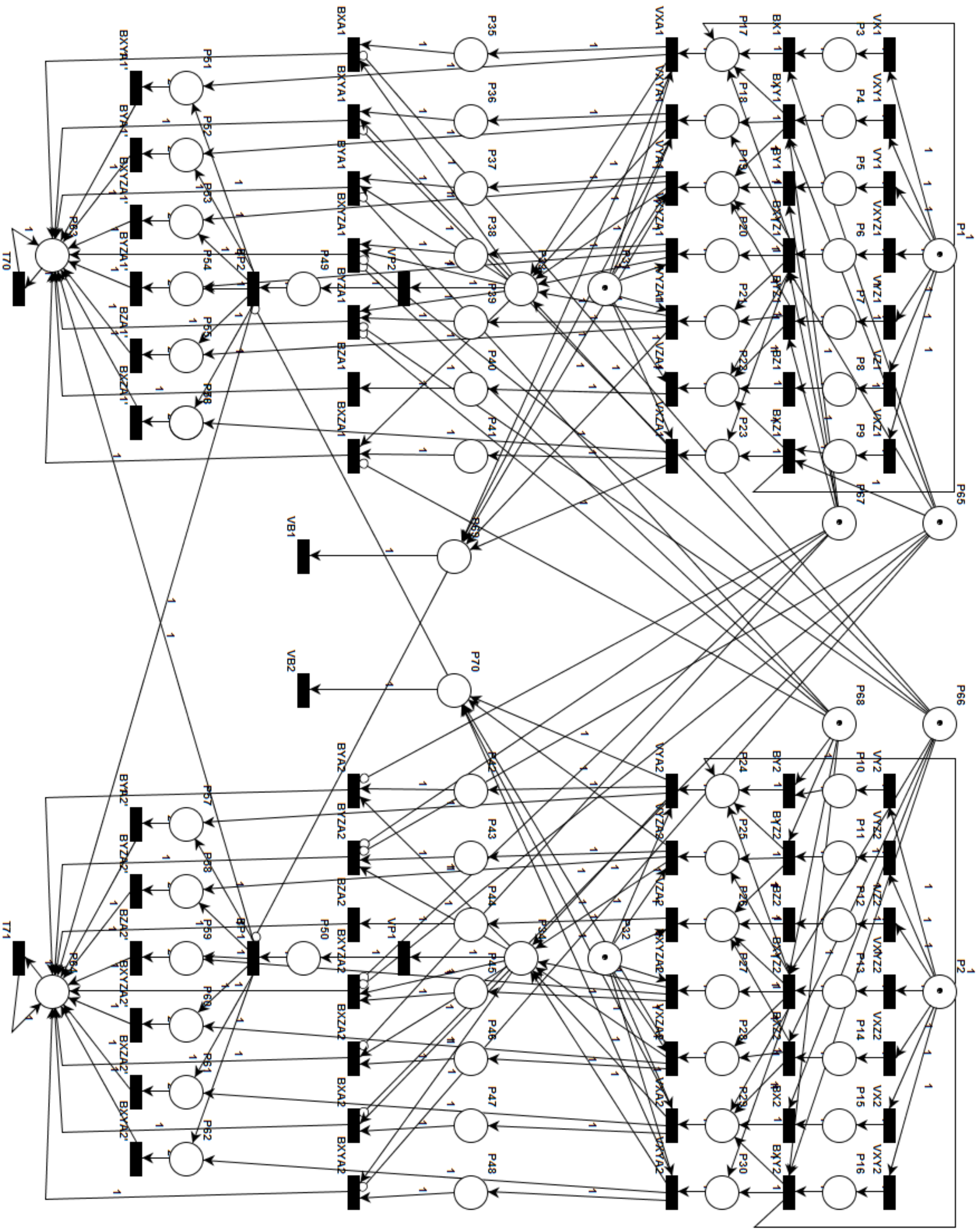


Figure 2.32: Deadlock-free hierarchical Petri net subpage for finite elements adjacent along Z axis.

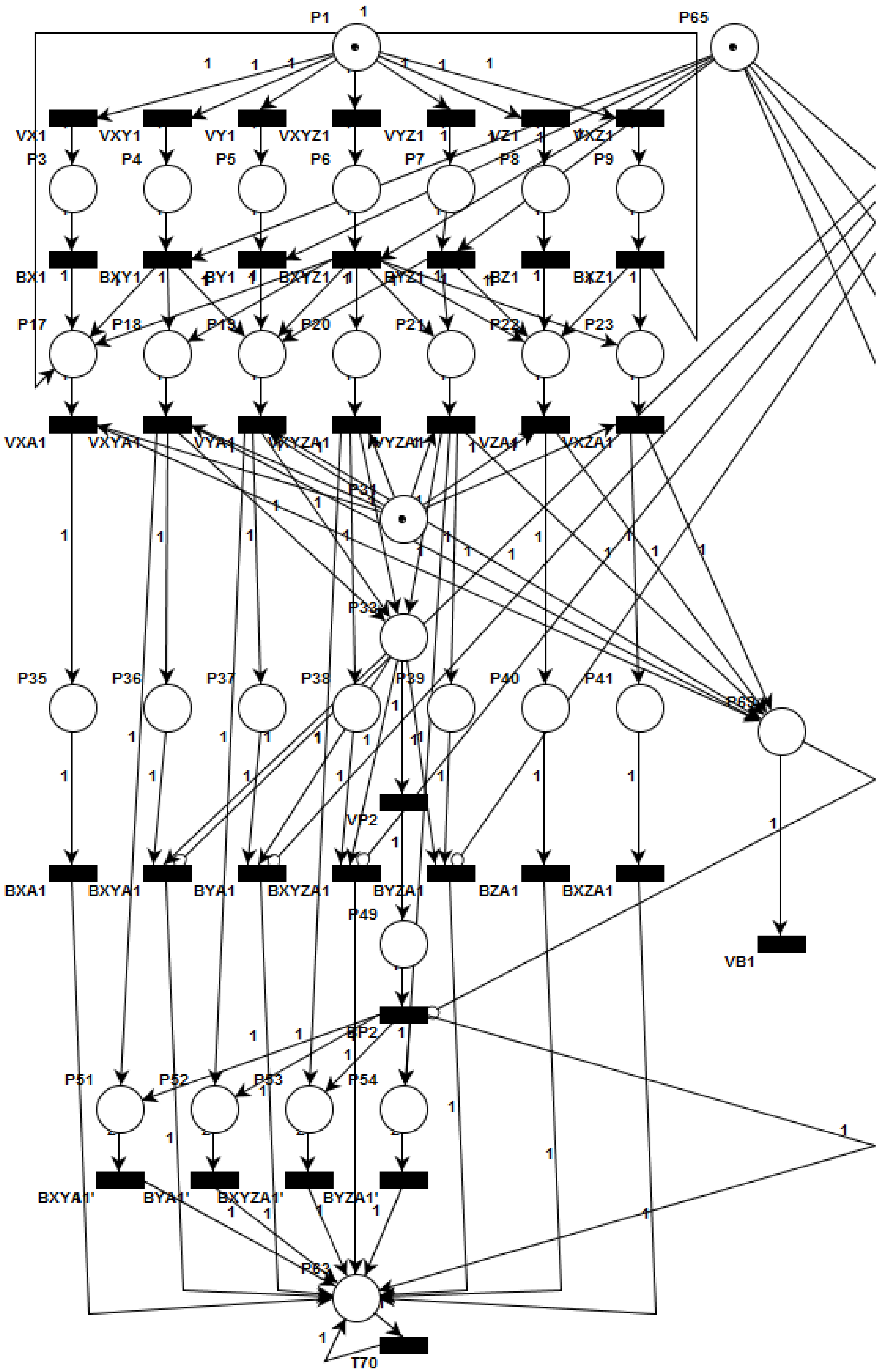


Figure 2.33: First part of deadlock-free hierarchical Petri net subpage for finite elements adjacent by edge.

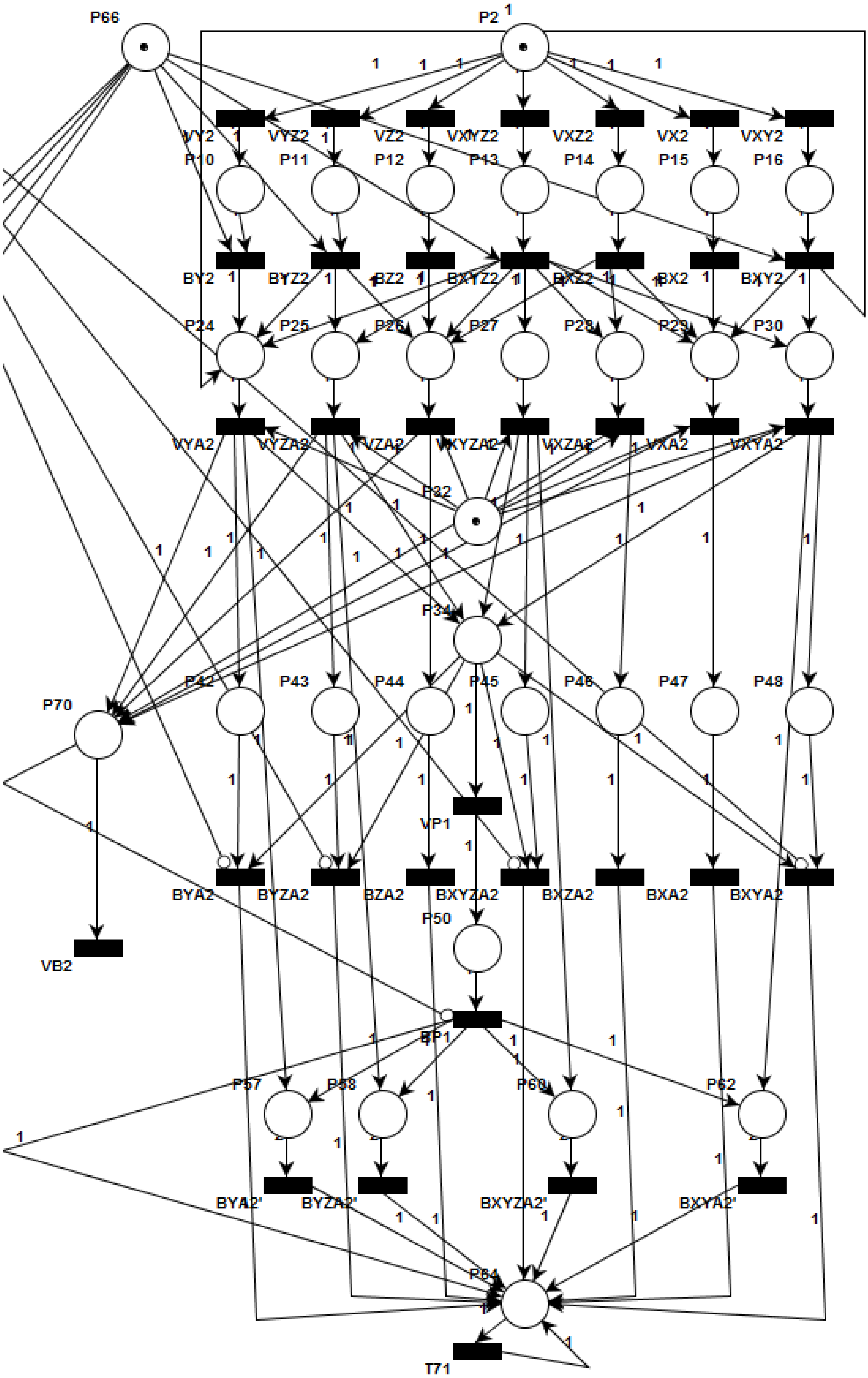


Figure 2.34: Second part of deadlock-free hierarchical Petri net subpage for finite elements adjacent by edge.

Chapter 3

Numerical results

3.1. 2D Example

The deadlock problem can be illustrated by simulating the two-dimensional (2D) magnetotelluric (MT) problem. This technique is used to determine the resistivity map of the Earth's subsurface by performing electromagnetic (EM) measurements. The main difference between MT and usual measurement acquisition scenarios is that MT uses natural sources generated within the ionesphere, and it does not require artificial sources. Thus, acquisition of MT measurements is rather inexpensive and it can cover large areas. Applications of MT measurements include hydrocarbon (oil and gas) exploration and finding suitable regions for storage of CO_2.

3.1.1. Strong formulation

Let us consider the 2D conductive media equation

$$\nabla \cdot \sigma \nabla u = \nabla \cdot \mathbf{J}^{imp}, \tag{3.1}$$

where σ is the conductivity of the media, u is the electric potential, and $\mathbf{J}^{imp}$ is the impressed electric current (the source). The above partial differential equation (PDE) is imposed on a computational domain $\Omega = [0,1]^d$, where d is the spatial dimension. Homogeneous Dirichlet boundary conditions are imposed

$$u = 0 \text{ on } \Gamma_D, \tag{3.2}$$

where $\Gamma_D = \partial\Omega$.

3.1.2. Weak formulation

The weak variational formulation is obtained by taking the L^2-scalar product with functions $v \in H^1_{\Gamma_D}(\Omega) = \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\}$, and integrating by parts to obtain:

$$\text{Find } u \in V = H^1_{\Gamma_D}(\Omega) \text{ such that} \tag{3.3}$$

$$b(v,u) = l(v), \forall v \in V, \tag{3.4}$$

where

$$b(v,u) = \int_\Omega \sigma \nabla v \cdot \nabla u dx, \text{ and} \tag{3.5}$$

$$l(v) = \int_\Omega v \cdot \mathbf{J}^{imp} dx \tag{3.6}$$

3.1.3. Deadlock problem

An hp-adaptive finite element method is employed for simulations of MT measurements [1]. The measurement acquisition scenario is considered, as described in Figure 3.1. It is composed of a source modeled as a plane wave operating at $100Hz$ coming from the top part of the domain, a layer of air, a background earth material with resistivity equal to $200\Omega m$, and three target zones with resistivities equal to $1000\Omega m$, $2000\Omega m$, and $50\Omega m$, respectively. This 2D model problem is governed by Maxwell's equations, and a Perfectly Matched Layer (PML) is incorporated in order to truncate the computational domain. Receivers are located along the air-earth interface. For simplicity, in this example only one receiver position on top of the $2000\Omega m$ resistivity layer is considered. For that position, the so-called hp goal-oriented adaptive strategy is employed [38].

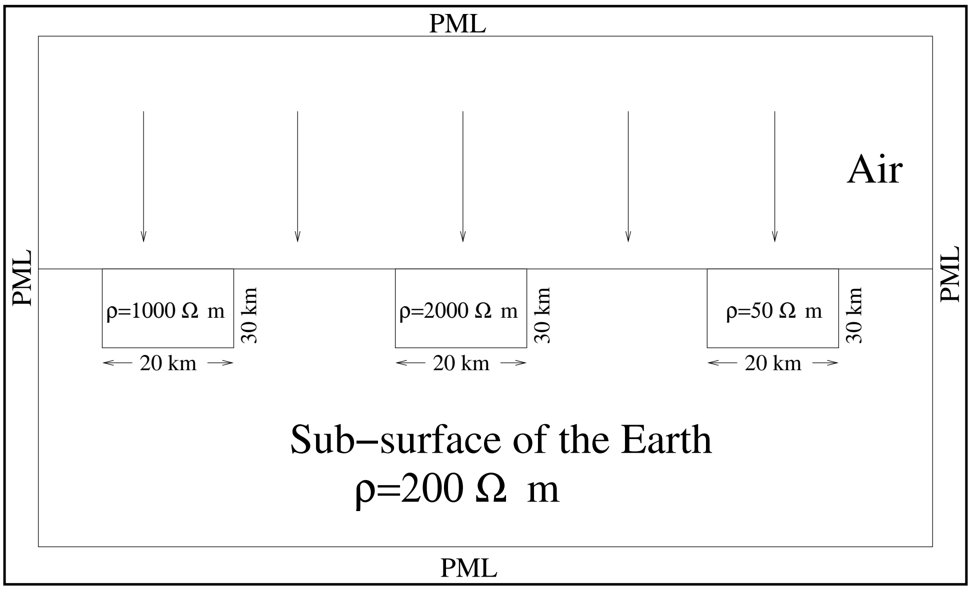


Figure 3.1: Geometry for the magnetolluric problem being solved

The original *hp2d* algorithm from [10] has been applied for the adaptive solution of this magnetolluric problem. The version without fixing the deadlock problem breaks down with just 6,245 unknowns, delivering an error of 0.03%, while for reliable analysis of MT results 0.001% accuracy is needed. The convergence history is presented in Figure 3.2. The solution obtained on the mesh with accuracy 0.03% where the deadlock happened is presented in Figure 3.3. It is clear that the accuracy is too low to estimate the properties of formation layers, since the solution over the part of domain representing the air is dominating. Figures 3.4 and 3.5 present the mesh with the deadlock problem.

A graph grammar model, expressing the mesh transformations performed by the *hp2d* code, was constructed. Next, the Petri net model was built for the analysis of the algorithm, and the deadlock problem was detected. Some additional graph grammar productions were added, eliminating the deadlock problem. The Petri nets model was constructed for the enhanced graph grammar and proved that the new model was indeed deadlock-free. Finally, the *hp2d* algorithm was corrected in order to continue the magnetolluric simulations without deadlock.

The computations were continued until the accuracy of 0.001% was reached, as required for the accurate magnetolluric problem solution. The final mesh delivering 0.001% accuracy is presented in Figure 3.6. The corresponding final solution is presented in Figure 3.7.

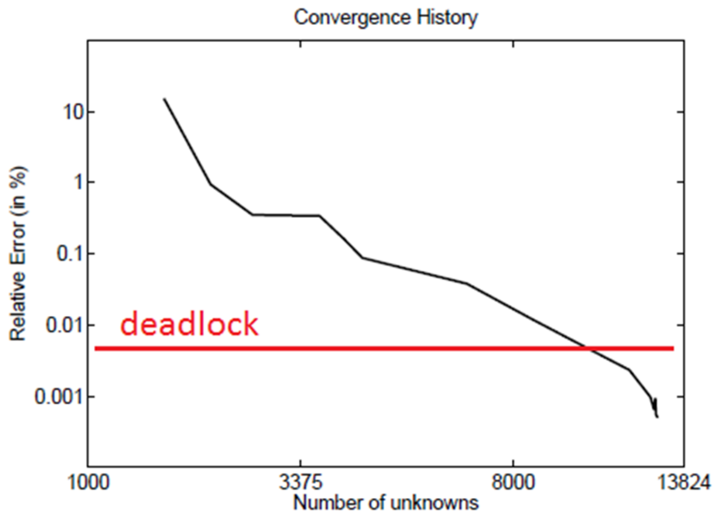


Figure 3.2: Convergence history for hp-adaptive finite element method simulations with the deadlock problem

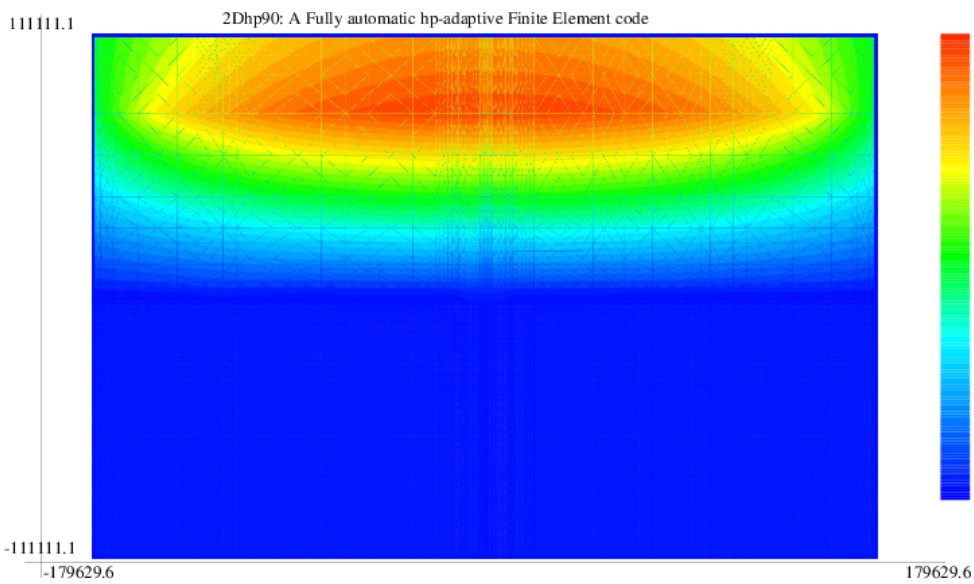


Figure 3.3: Solution over the mesh with 0.03% accuracy where the deadlock problem occured

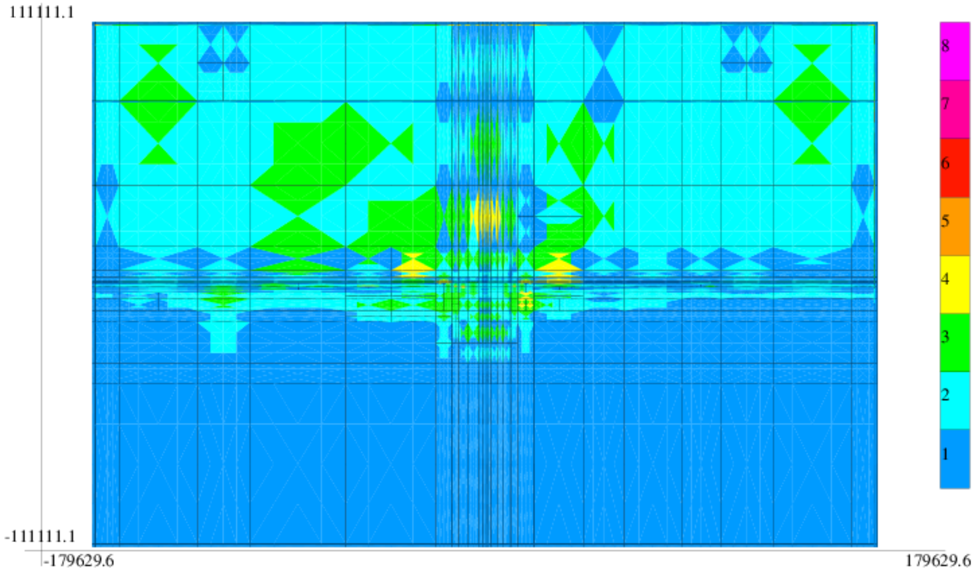


Figure 3.4: Global view on the hp-refined mesh with deadlock

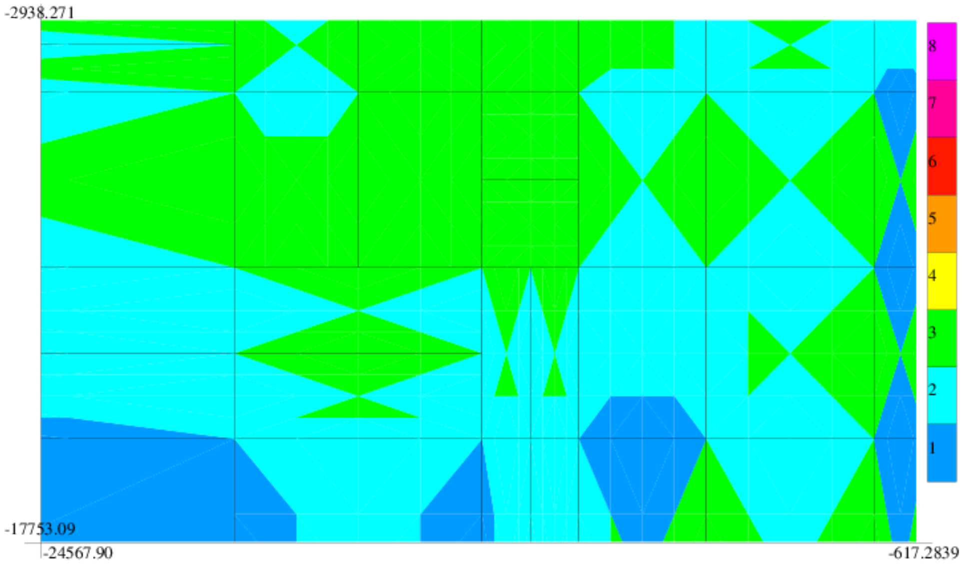


Figure 3.5: Amplification with the factor of 100 towards the deadlock area

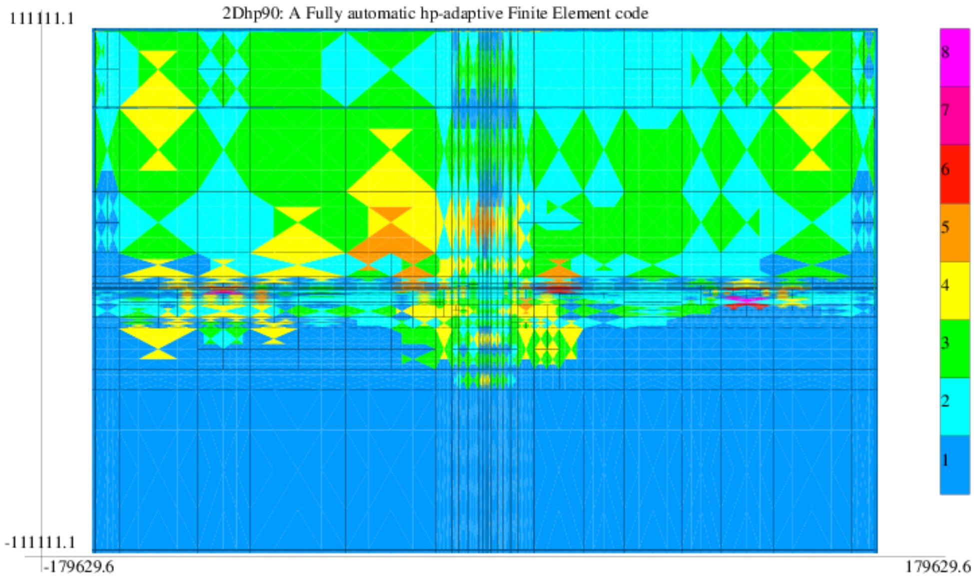


Figure 3.6: Global view of the final hp-refined mesh without deadlock

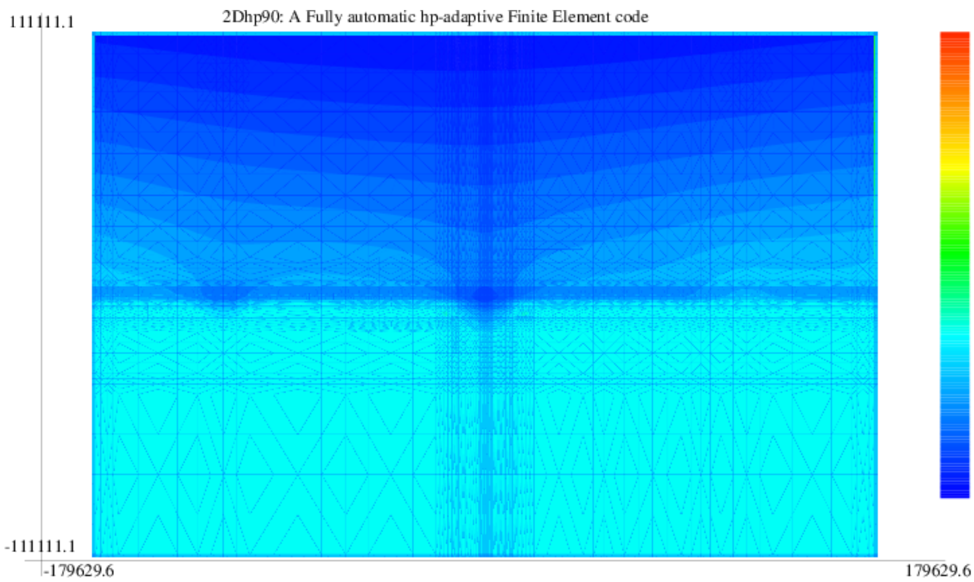


Figure 3.7: Solution over the final mesh with 0.001% accuracy

3.2. 3D Example

The *hp3d* code was executed with the original mesh adaptation algorithm over a problem consisting in the simulation of 3D direct current (DC) borehole resistivity measurements in deviated wells. A quantity of interest, in this case the voltage, is measured at a receiver electrode located in a borehole logging instrument.

As the logging instrument moves along the borehole, the voltage measured at the receivers is expected to be proportional to the electrical resistivity of the nearby formation. Thus, logging instruments are used to estimate the properties (in this case, the electrical conductivity or resistivity) of the sub-surface material, with the ultimate objective of describing hydrocarbon (oil and gas) bearing formations.

In this section, the behavior of a resistivity logging instrument is simulated by performing computer-based simulations of resistivity logging instruments in a borehole environment [38].

Of particular interest to the oil industry are 3D simulations of resistivity measurements in deviated wells, where the angle between the borehole and the formation layers is not equal to 90 degrees.

Different electrode configurations (see Figure 3.8) and dip angles (which is the angle of incidence between the well and the formation layers) are considered.

3.2.1. Strong formulation

Find $u : \Omega \ni x \to u(x) \in R$ where $\Omega \subset R^3$ the electrostatic scalar potential such that

$$-div\left(\sigma\nabla u\right) = \nabla \cdot J \text{ in } \Omega, \tag{3.7}$$

(the conductive media equation), where $\nabla \cdot J$ is the load (divergence of the impressed current) and σ represents the conductivity of the media, defined according to Figure 3.9. The electrostatic scalar potential u is related to the electric field E by

$$E = -\nabla u. \tag{3.8}$$

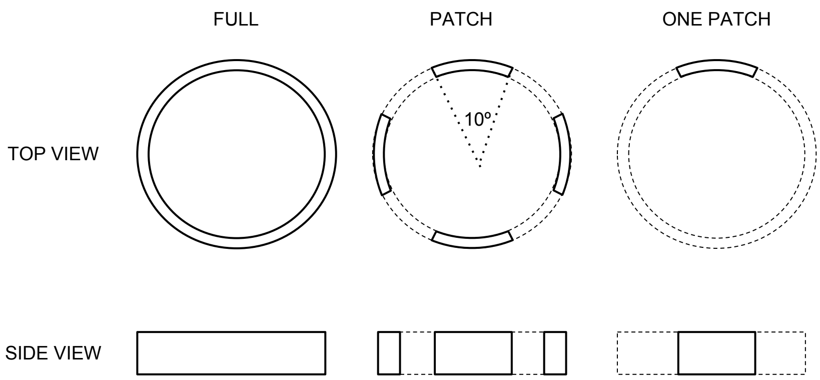


Figure 3.8: The geometry of antennas.

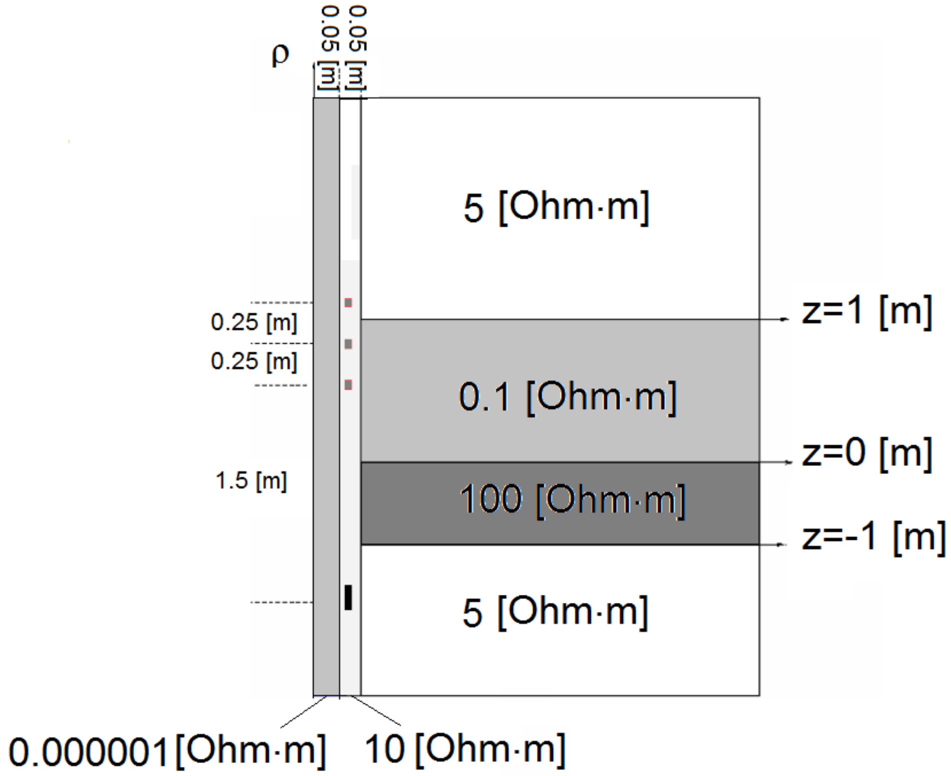


Figure 3.9: The conductivities of the borehole, mandrel and formation layers in cylindrical coordinates.

The boundary conditions are defined as

$$u = 0 \text{ on } \partial\Omega. \tag{3.9}$$

3.2.2. Weak formulation

The strong formulation is transformed into the weak one: Find $u \in V$ such that

$$b(u, v) = l(v) \quad \forall v \in V \tag{3.10}$$

$$b(u, v) = \int_\Omega \sum_{i=1}^{3} \sigma \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_i} dx \tag{3.11}$$

$$l(v) = \int_\Omega \sum_{i=1}^{3} v \frac{\partial J}{\partial x_i} dx \tag{3.12}$$

where $V = H_0^1(\Omega)$.

3.2.3. Deadlock problem

The adaptive algorithm from the *hp3d* code [11] generates a sequence of computational grids delivering exponential convergence of the numerical error with respect to the mesh size. The sequence of meshes is obtained by performing h refinements (by breaking selected elements in one of eight possible ways) or p refinements (by modifying the polynomial order of approximation on finite element edges, faces, and interiors). This adaptation process is performed by considering a sequence of coarse and fine grids, and by selecting the optimal refinements over the coarse grid resulting from comparison of the coarse and fine grid solutions, compare Figure 3.10. For the detailed description of the algorithm selecting the optimal refinement, please refer to [40].

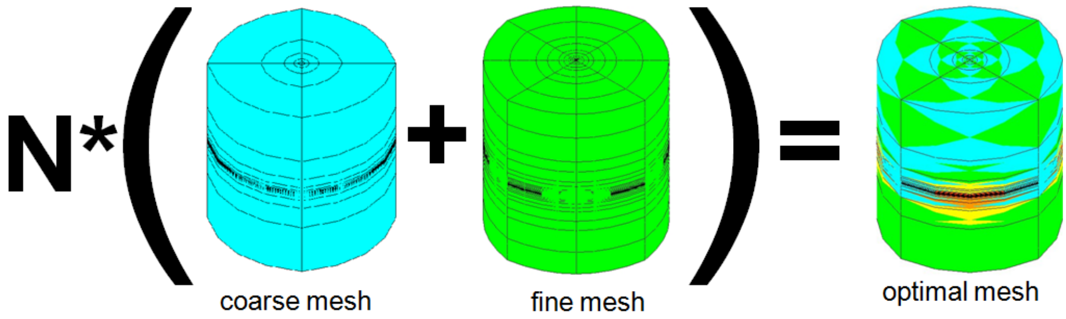


Figure 3.10: The sequence of coarse, fine and optimal grids generated by the self-adaptive hp-FEM algorithm.

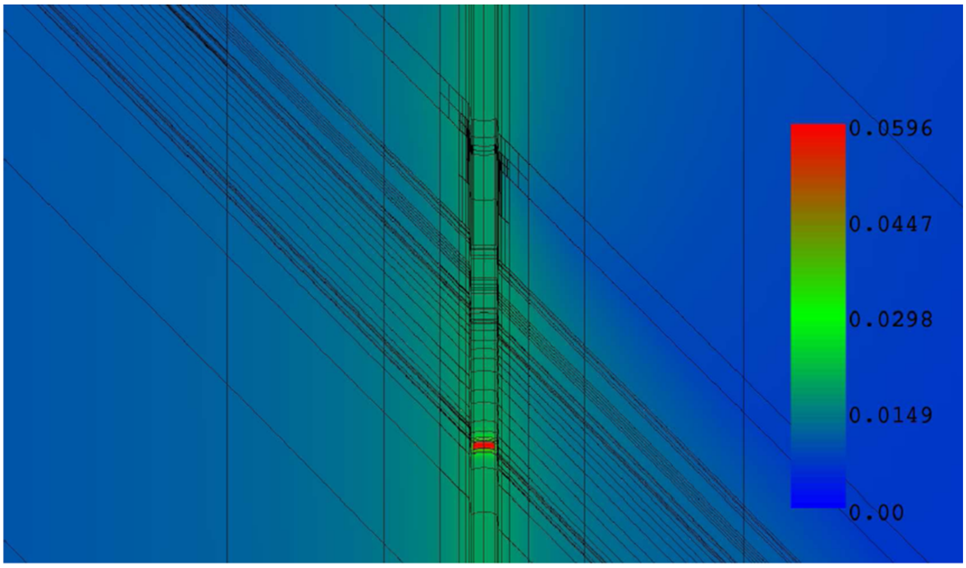


Figure 3.11: Exemplary resulting potential at the receiver electrode for a single position of a logging tool for 60 degrees deviated well.

The original mesh adaptation algorithm from *hp3d* code stopped after executing several h refinements due to adaptation deadlock. In order to analyze the problem, the graph grammar model of the adaptation algorithm was constructed, as summarized in Figure 2.11. Afterwards, the hierarchical Petri net model was constructed, based on the nets presented in Figures 2.20-2.25, according to Algorithm 2.2.2. The hierarchical Petri net detected a deadlock scenario, meaning the adaptation algorithm from [11] needed corrections. To this end, the graph grammar was extended with additional productions presented in Figure 2.28. Subsequently, the hierarchical Petri net based on the augmented nets presented in Figures 2.29-2.34, reflecting the extended graph grammar model, was constructed according to Algorithm 2.2.2. The new hierarchical Petri net was live (no dead state was possible), which allowed to correct the original adaptation algorithm from *hp3d* and to overcome the deadlock problem and complete the computation process.

The problem of propagation of electromagnetic waves has been solved for a sequence of positions corresponding to different locations of the logging tool moving along the borehole, for axial-symmetric as well as 30 and 60 degrees deviated well. The exemplary resulting potential at the receiver electrode for a single position of the tool is presented in Figure 3.11. In this picture a 2D vertical cross-section of the 3D mesh was considered. The cross-section is going through the borehole with receiver electrodes.

The resulting logging curves for different dip angles and different kind of antennas are presented in Figures 3.12-3.14.

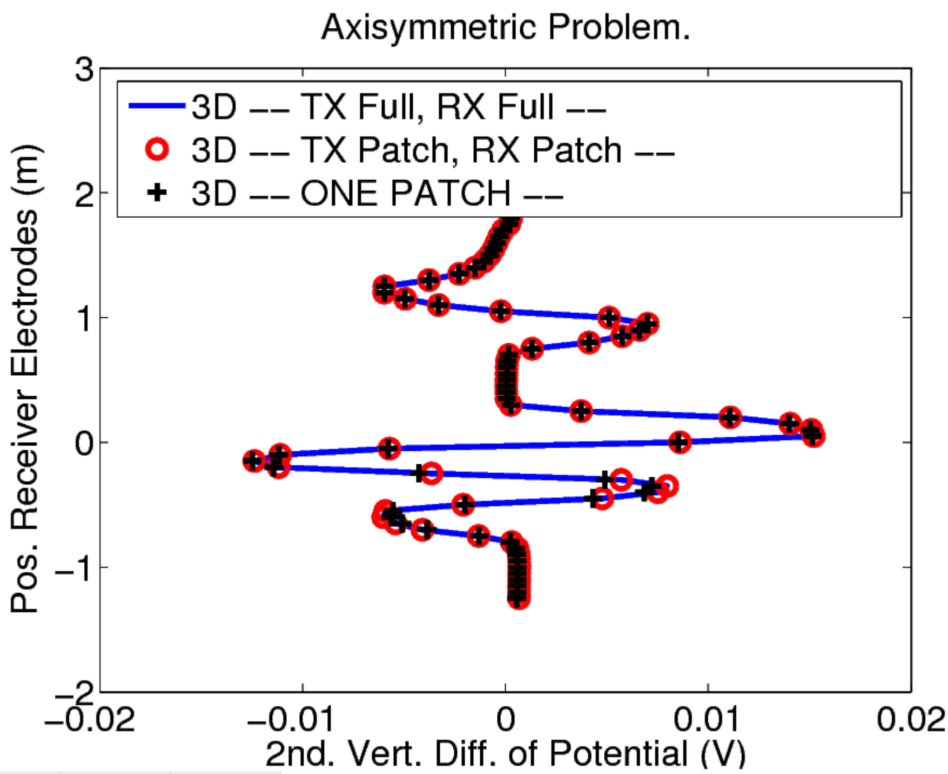


Figure 3.12: Logging curves for different antennas for axial-symmetric case.

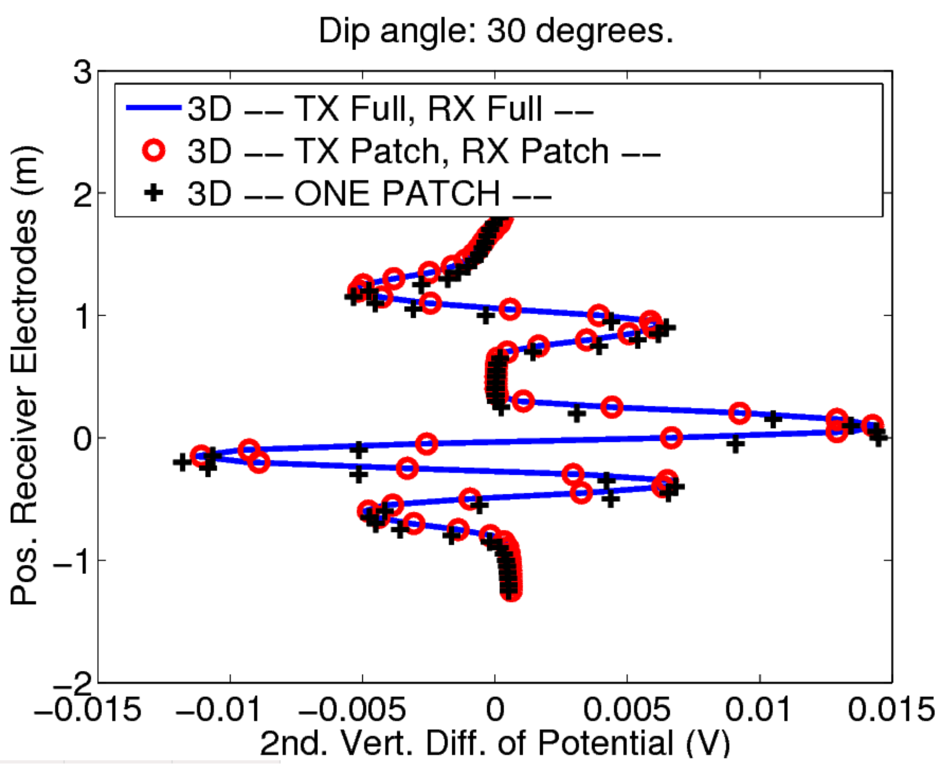


Figure 3.13: Logging curves for different antennas for 30 degrees deviated well case.

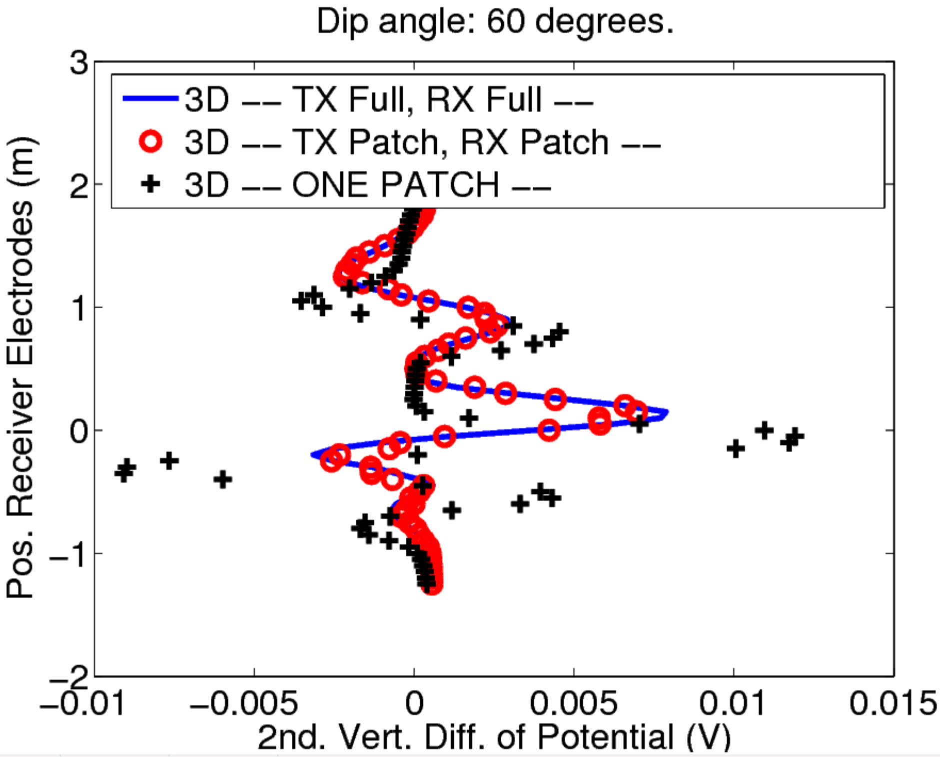


Figure 3.14: Logging curves for different antennas for 60 degrees deviated well case.

From the physical point of view, the following can be observed:

- For the axisymmetric case (Figure 3.12), a response independent of the type of antenna is obtained, as physically expected;
- As the dip angle increases (Figures 3.13 and 3.14), the effect of the antenna becomes noticeable, the "one-patch antenna" being the one that is more sensitive to variations in the formation resistivity.

Chapter 4

Conclusions and future remarks

This thesis introduced a Petri net model for two- and three-dimensional anisotropic mesh adaptation algorithms. The two-dimensional model was based on the adaptive algorithm as implemented in *hp2d* code, described in [10]. Similarly, the three-dimensional model was based on the adaptive algorithm as implemented in *hp3d* code, described in [11]. Both two- and three-dimensional models were based on a set of graph transformations, called graph grammar productions, that described the mesh transformations performed by adaptive algorithms from *hp2d* and *hp3d* codes, respectively. The Petri net model allowed to analyze the correctness of the mesh adaptation algorithms.

The Petri net model presented in this thesis allowed to detect potential deadlocks that may happen for specific configurations of mesh refinements in both two- and three-dimensional models. The possibility of deadlock was proved by providing a sequence of fired transitions (executed graph grammar productions) leading to a dead state of the hierarchical Petri net subpage. PIPE software was used to execute the simulation of the part of the Petri net model.

The deadlock problems were overcome by introducing some additional graph grammar productions, that allowed for an upgrade of some previous anisotropic mesh refinements to new isotropic ones. The enhanced graph grammar model expressing the new transformations was analyzed again by using the Petri net model. The reachability graph was constructed in PIPE tool, and the automatic analysis showed that the enhanced model was deadlock-free. This in turn allowed for modification of both *hp2d* and *hp3d* codes so they became deadlock-free.

The model described in this thesis was successfully employed to remove the deadlock from mesh adaptation algorithms applied for solution of two challenging engineering problems. The first one was the two-dimensional simulation of the magnetotelluric measurements performed in order to identify oil and gas location in the ground. The second one was three-dimensional direct current borehole resistivity measurement simulations in deviated wells, also performed in order to find carbohydrants bearing formations. These problems are of great importance to the geophysical community.

An obvious direction for future research in this field is modeling other shapes of computational meshes, for which anisotropic adaptation may lead to deadlock scenarios, e.g. two-dimensional meshes with triangular elements, two-dimensional meshes with mixed rectangular and triangular elements, two-dimensional unstructured grids, three-dimensional meshes with mixed tetrahedral elements or three-dimensional meshes with mixed tetrahedral, hexehedral, prism and pyramid elements.

Appendix A

Exemaplary analysis of Petri nets modeling 2D finite element method

The appendix presents exemplary analysis with two Petri nets. The first one concerns the “vertical” pair of elements subject to mesh refinements as expressed by the *hp2d* algorithm [10]. This analysis leads to a dead state. The particular steps are presented in Figures A.1-A.9. The second one concerns the “vertical” pair of elements subject to mesh refinements as expressed by the corrected mesh adaptation algorithm, deadlock-free. The particular steps are presented in Figures A.10-A.20.

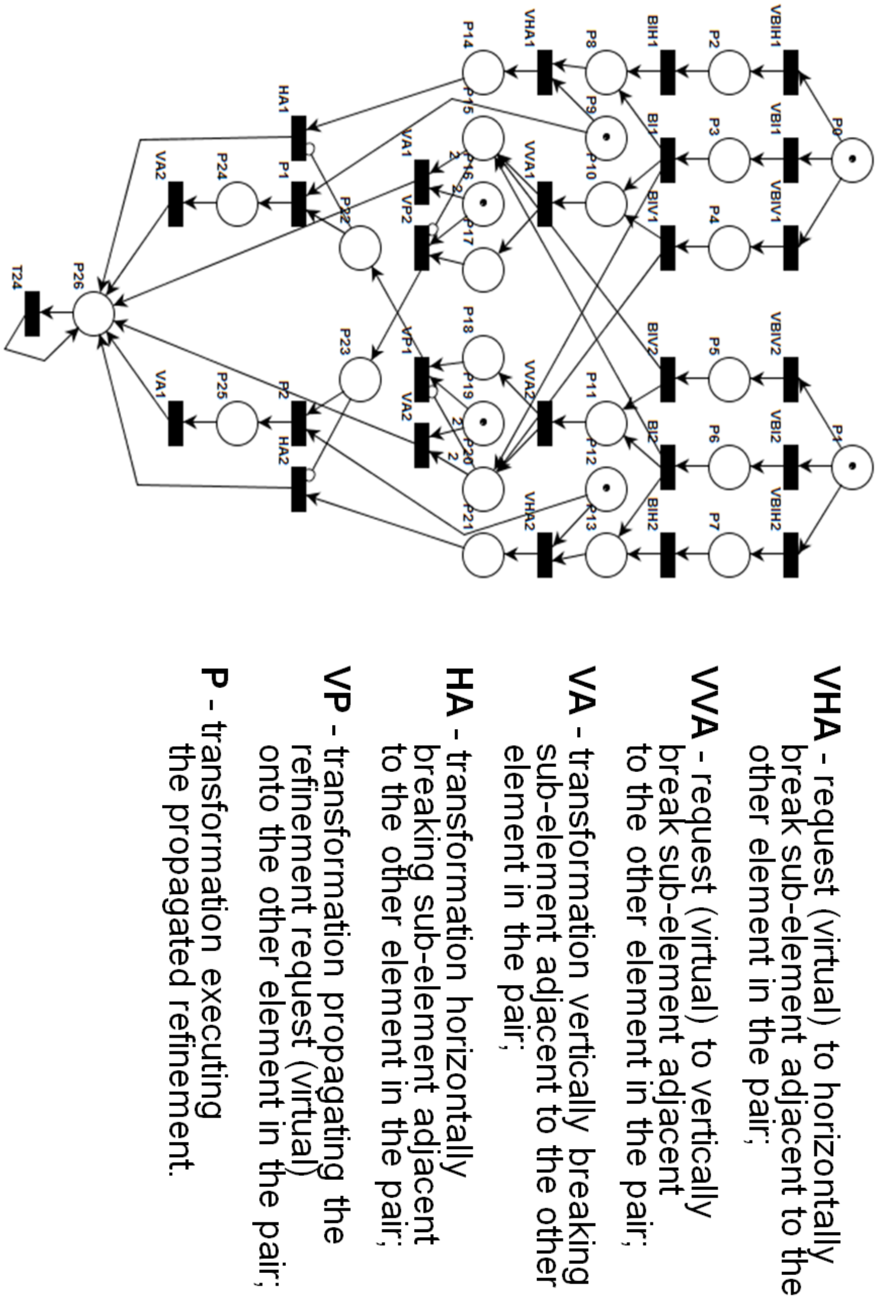


Figure A.1: Deadlock detection. The initial state

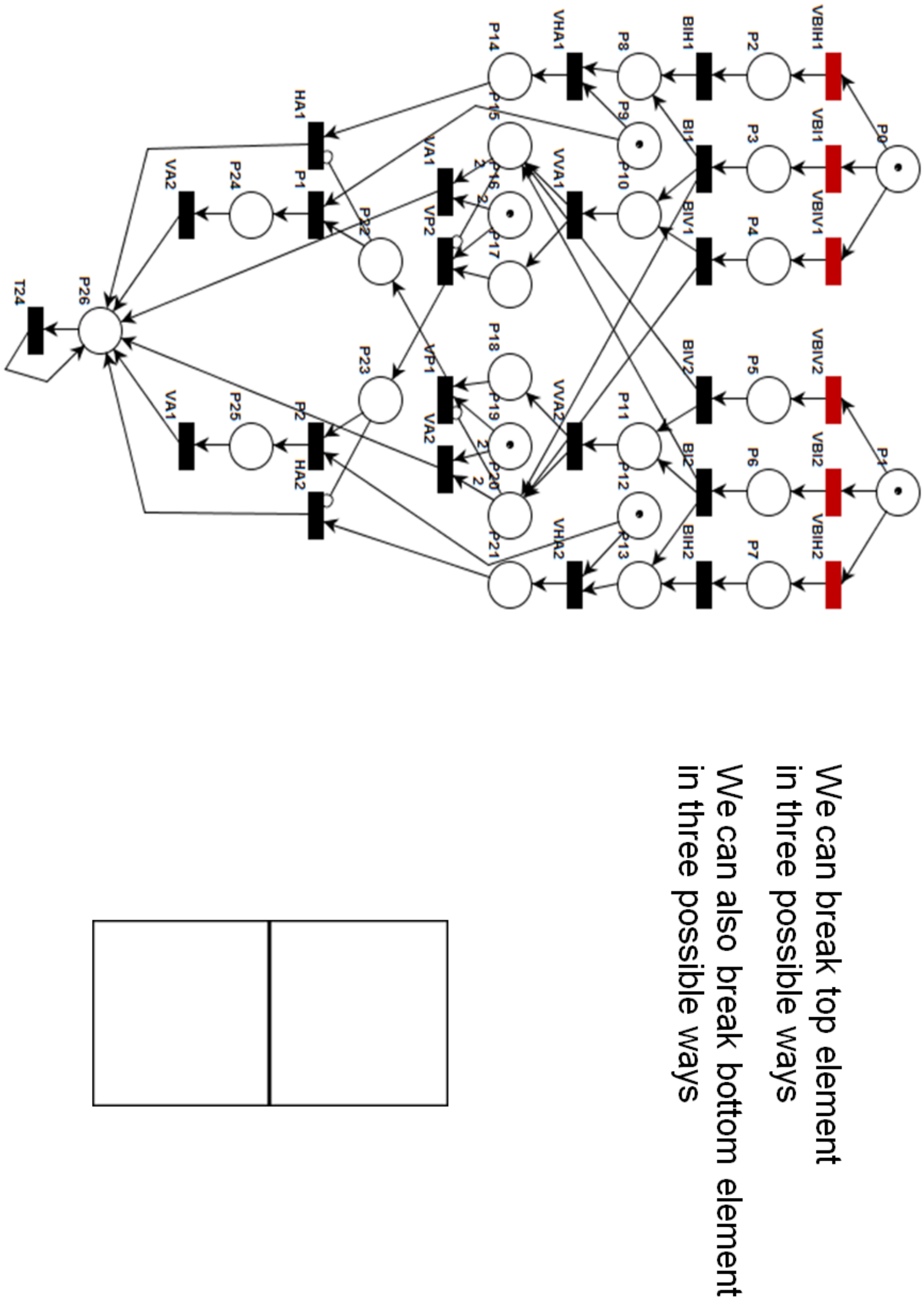


Figure A.2: Deadlock detection. Stage 1

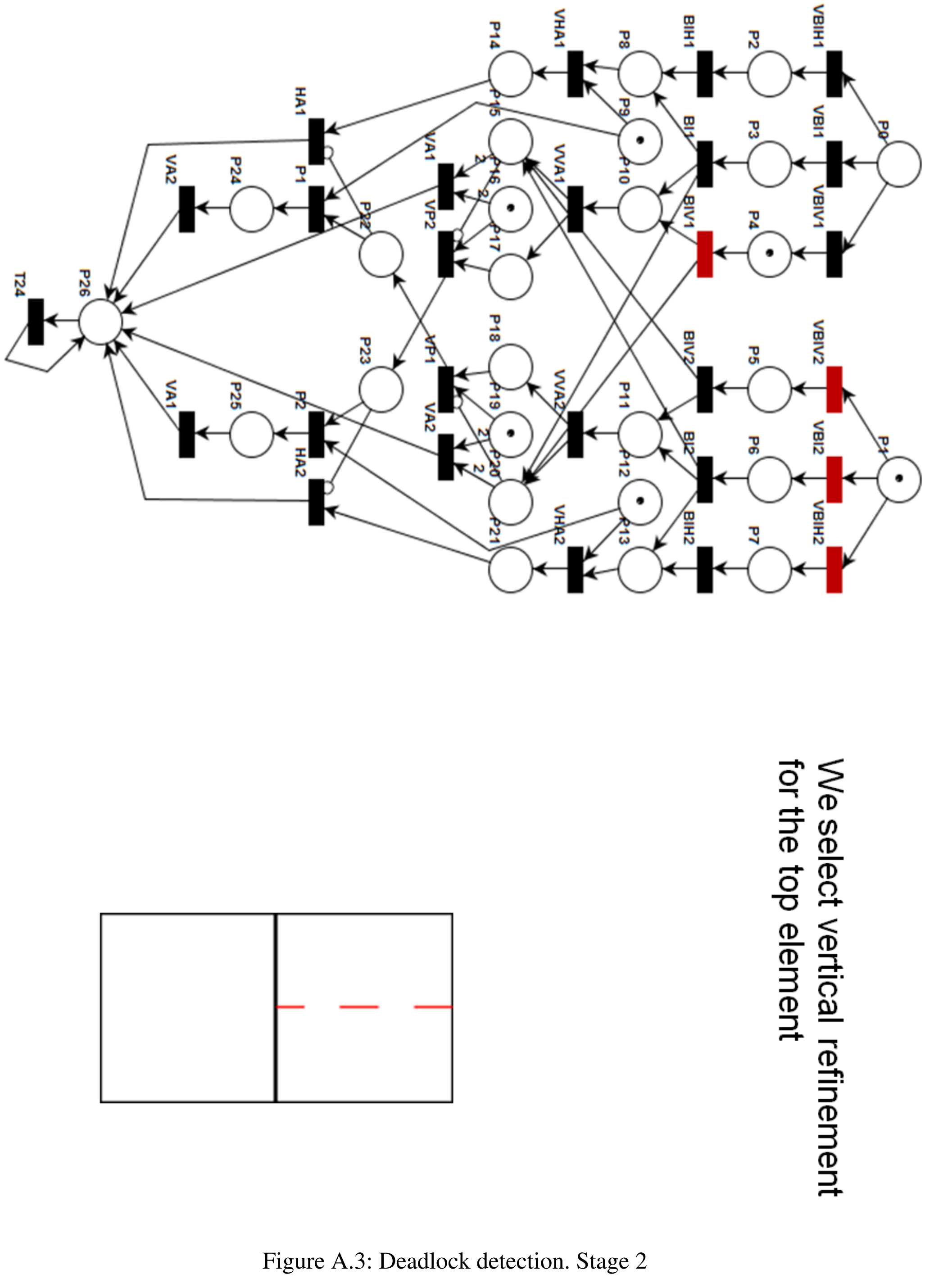


Figure A.3: Deadlock detection. Stage 2

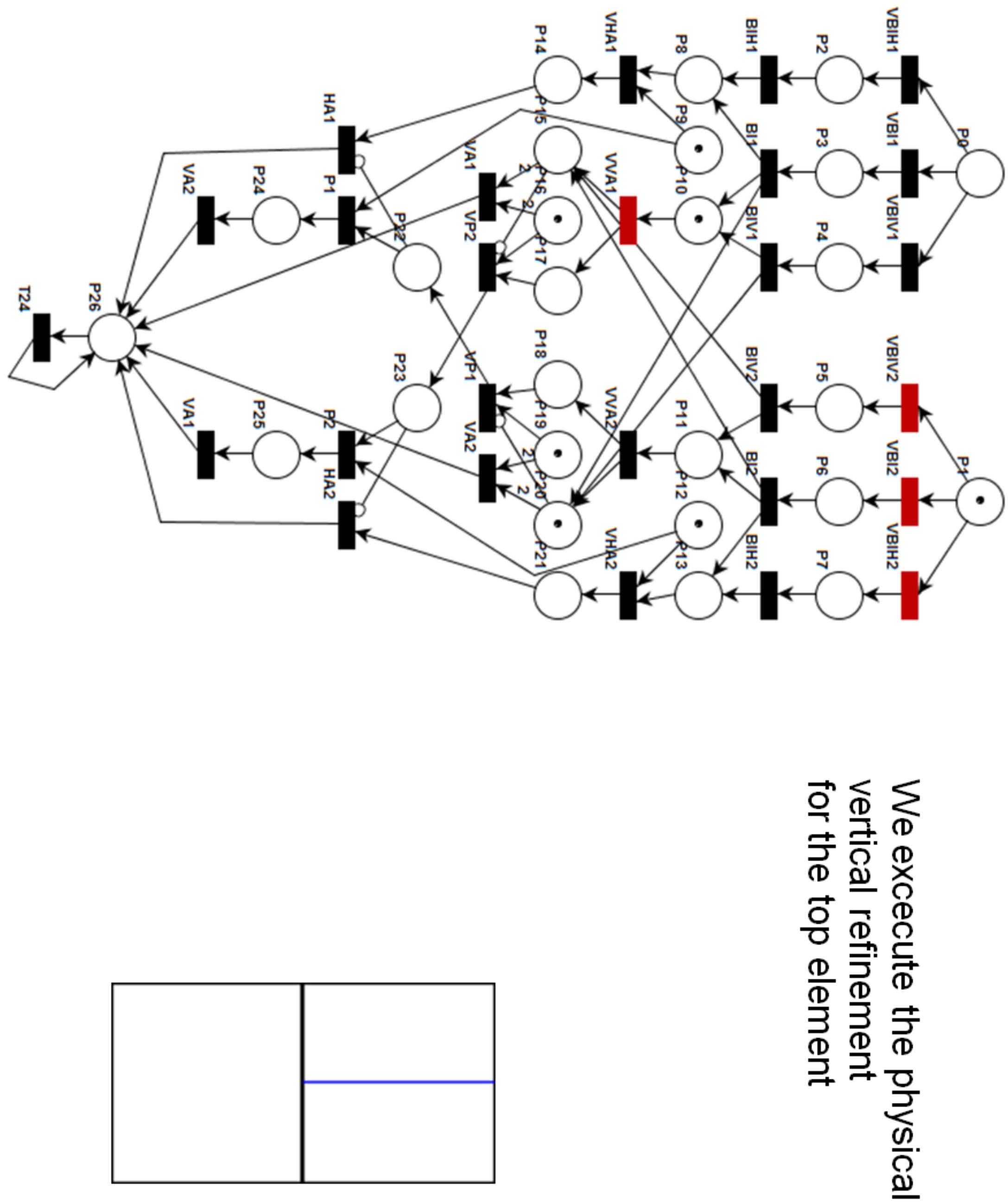


Figure A.4: Deadlock detection. Stage 3

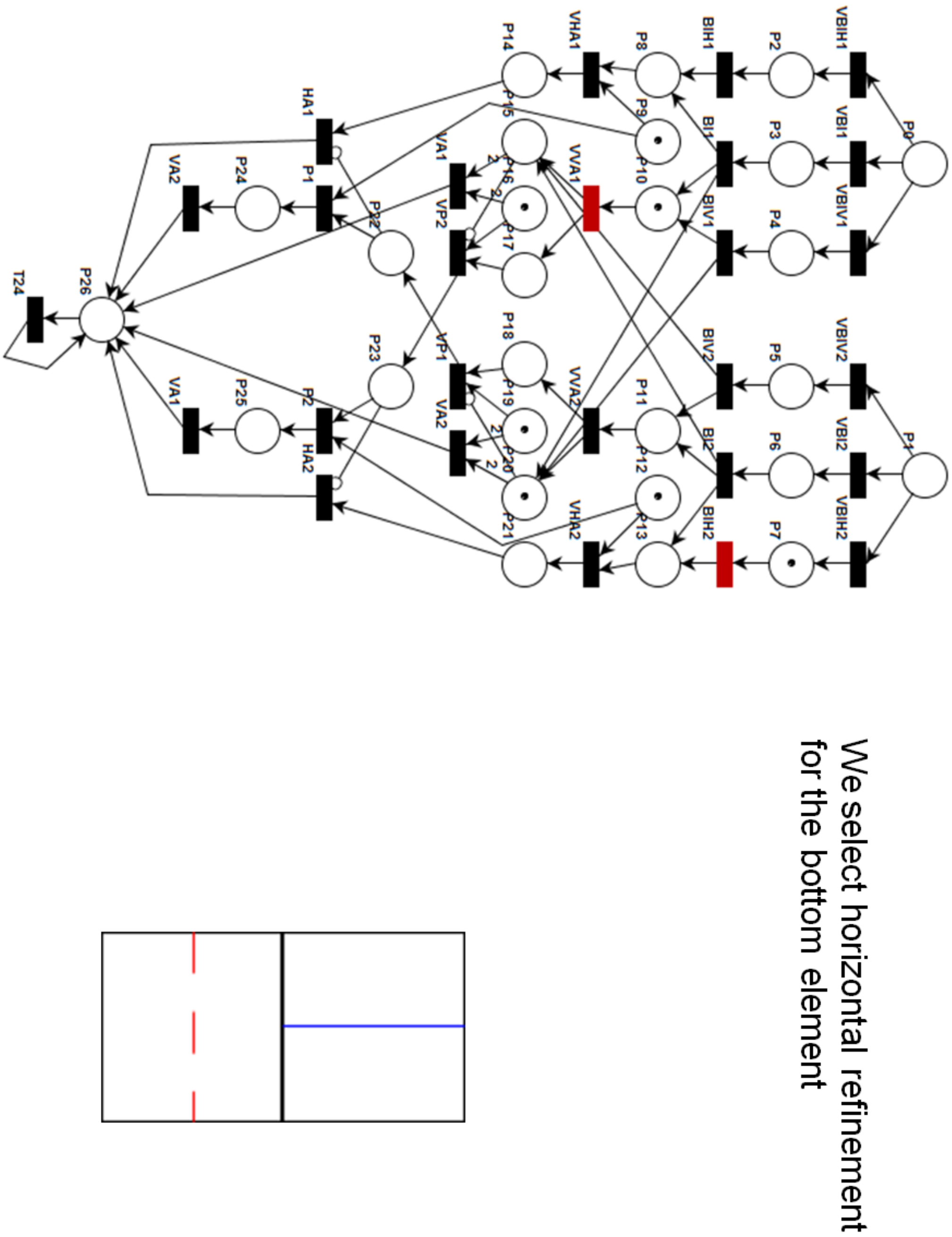


Figure A.5: Deadlock detection. Stage 4

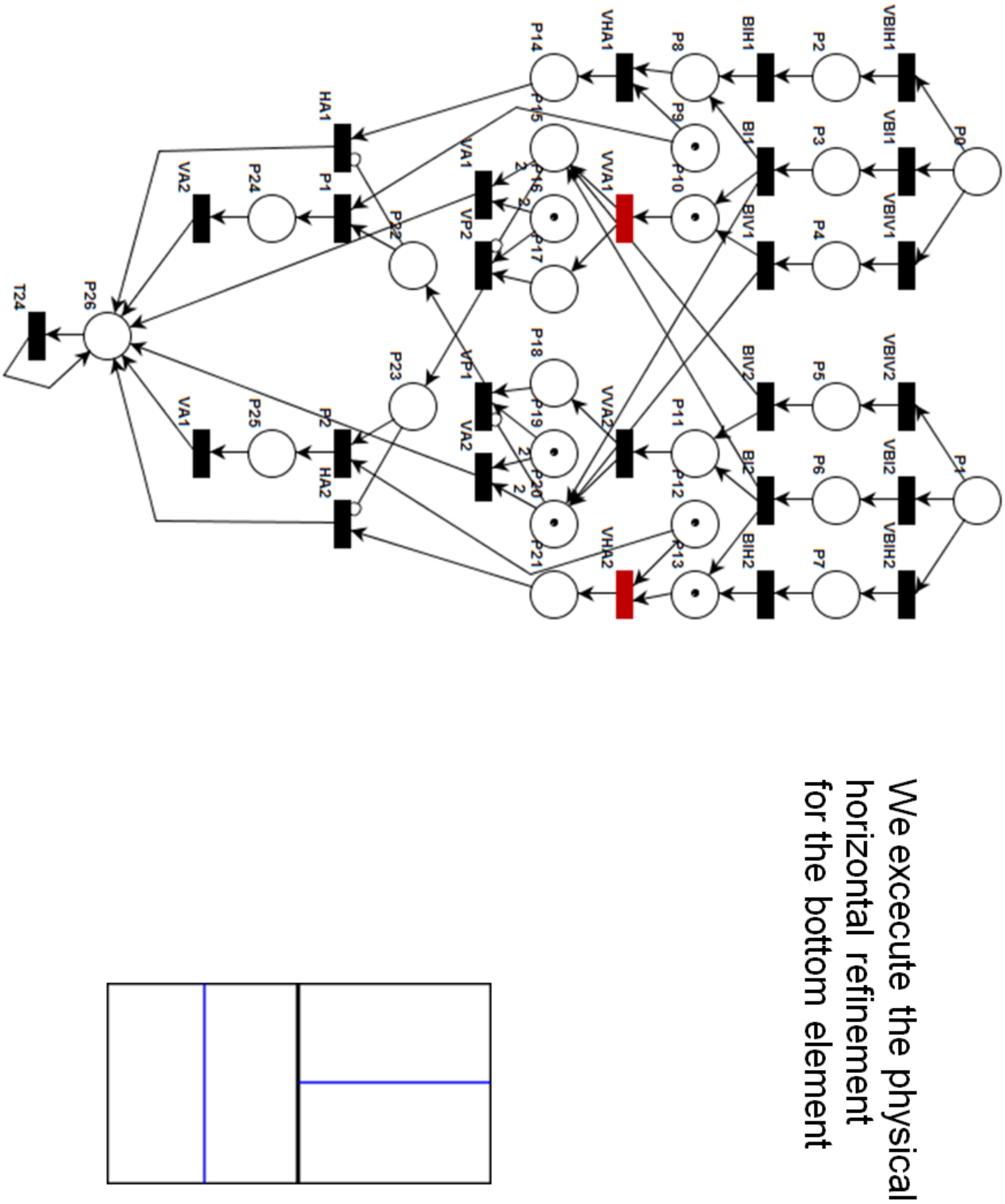


Figure A.6: Deadlock detection. Stage 5

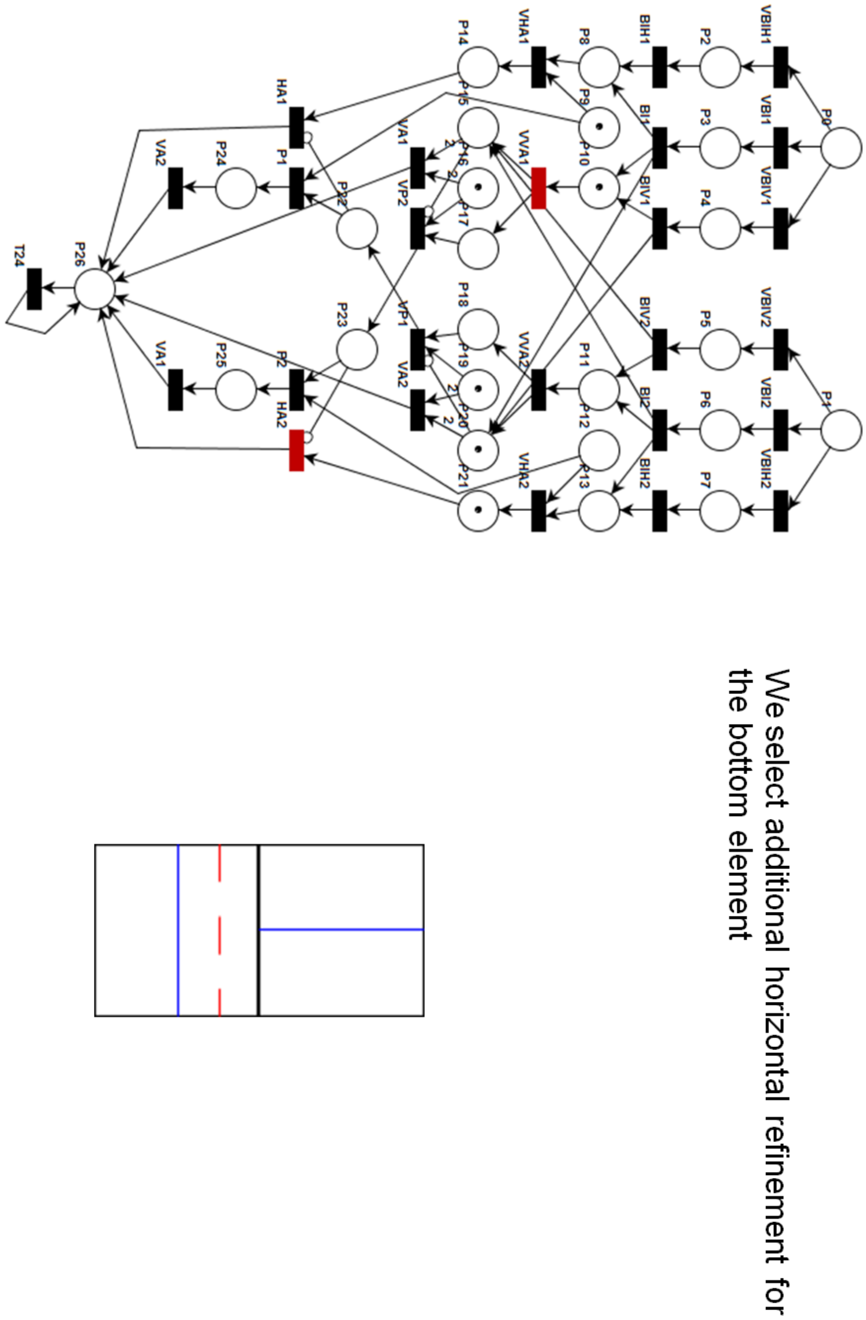


Figure A.7: Deadlock detection. Stage 6

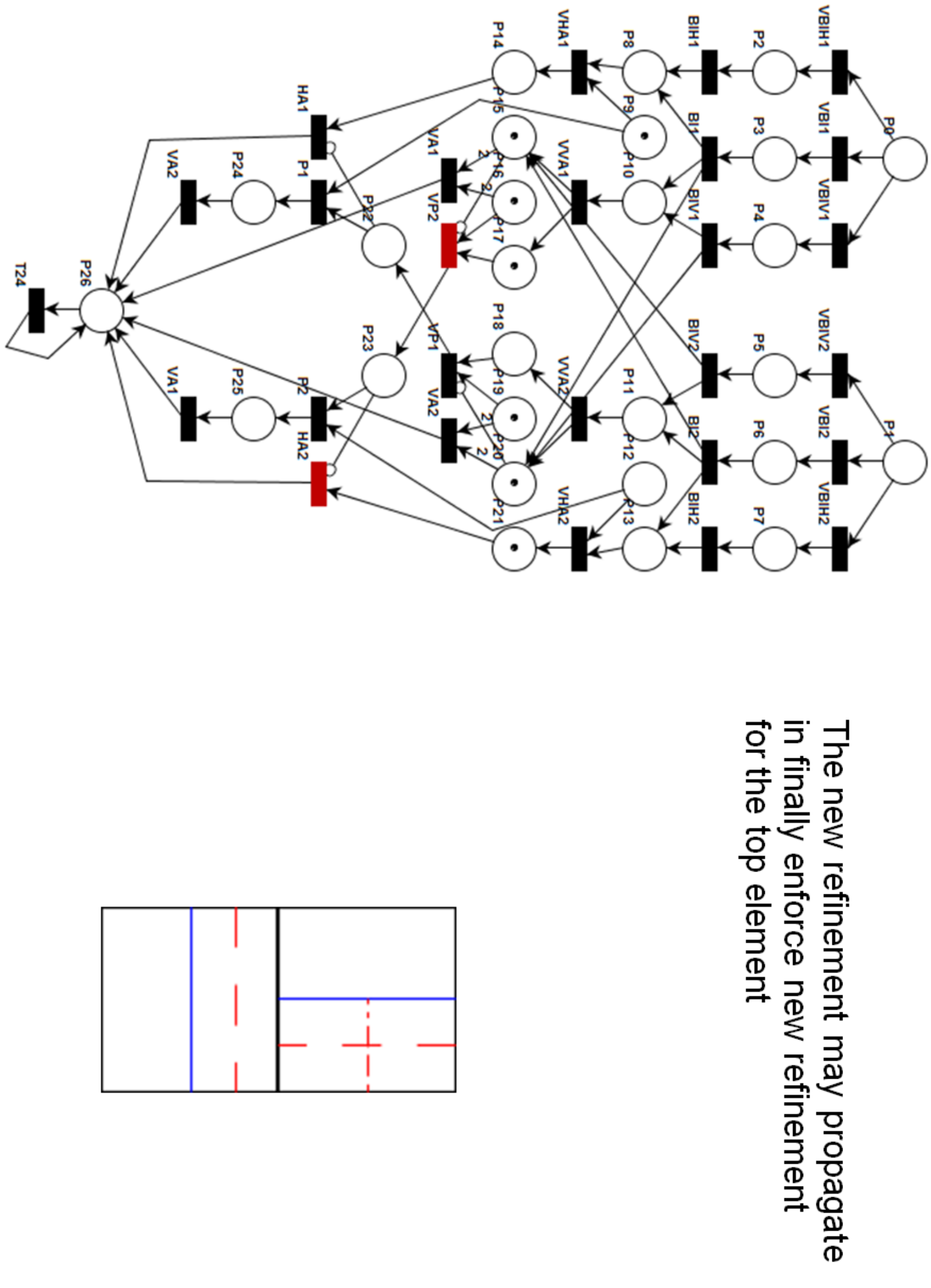


Figure A.8: Deadlock detection. Stage 7

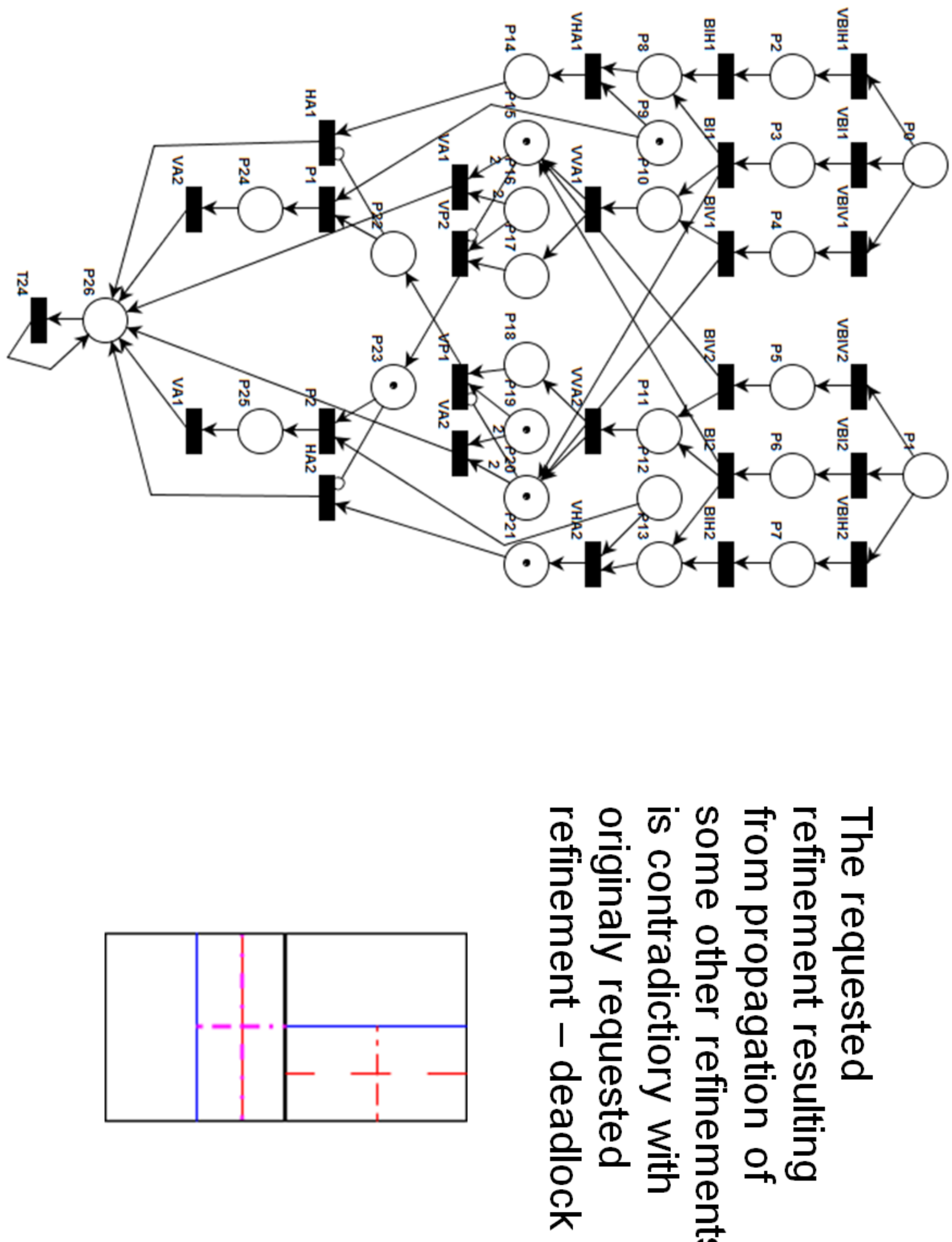


Figure A.9: Deadlock detection. Stage 8

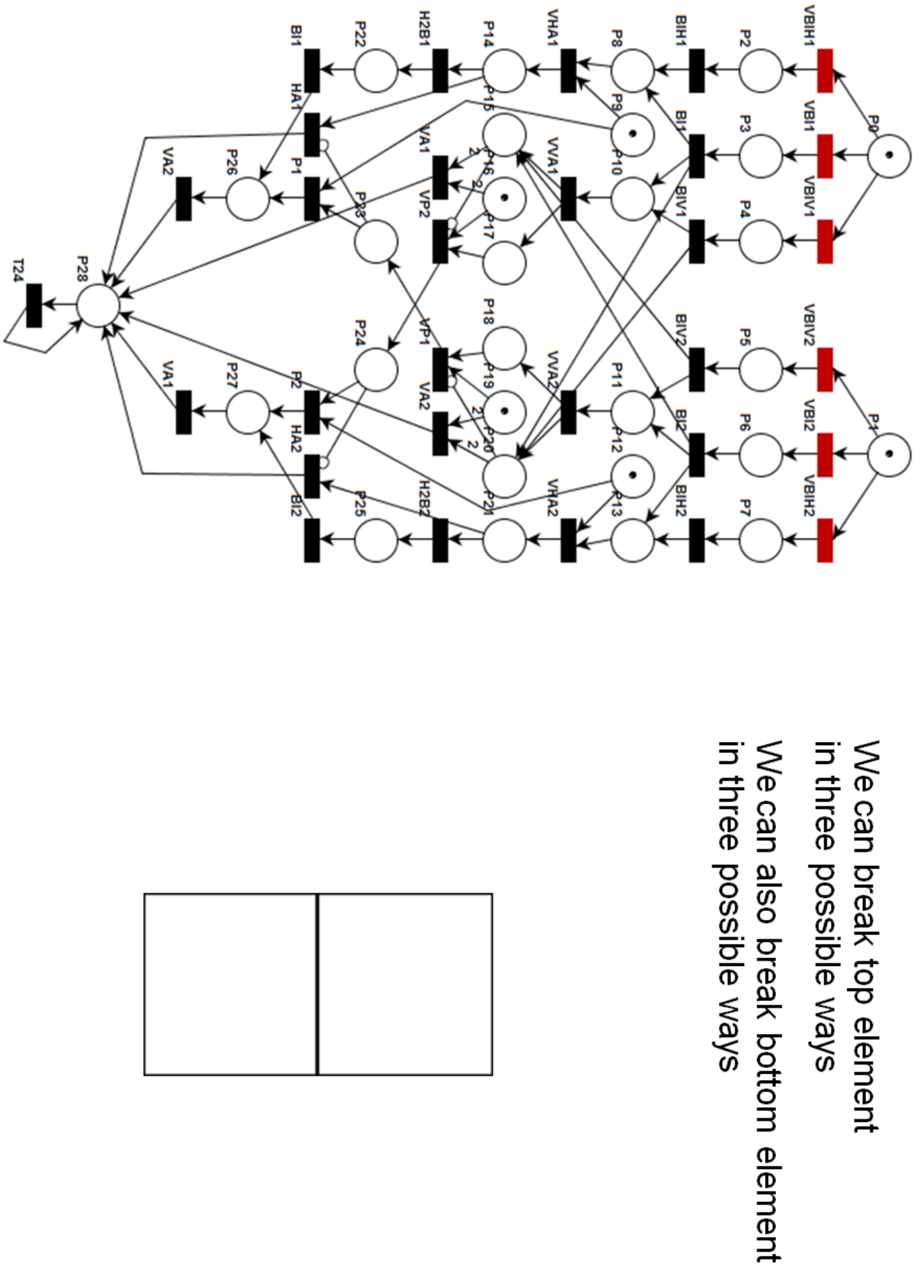


Figure A.10: Deadlock-free case. The initial state

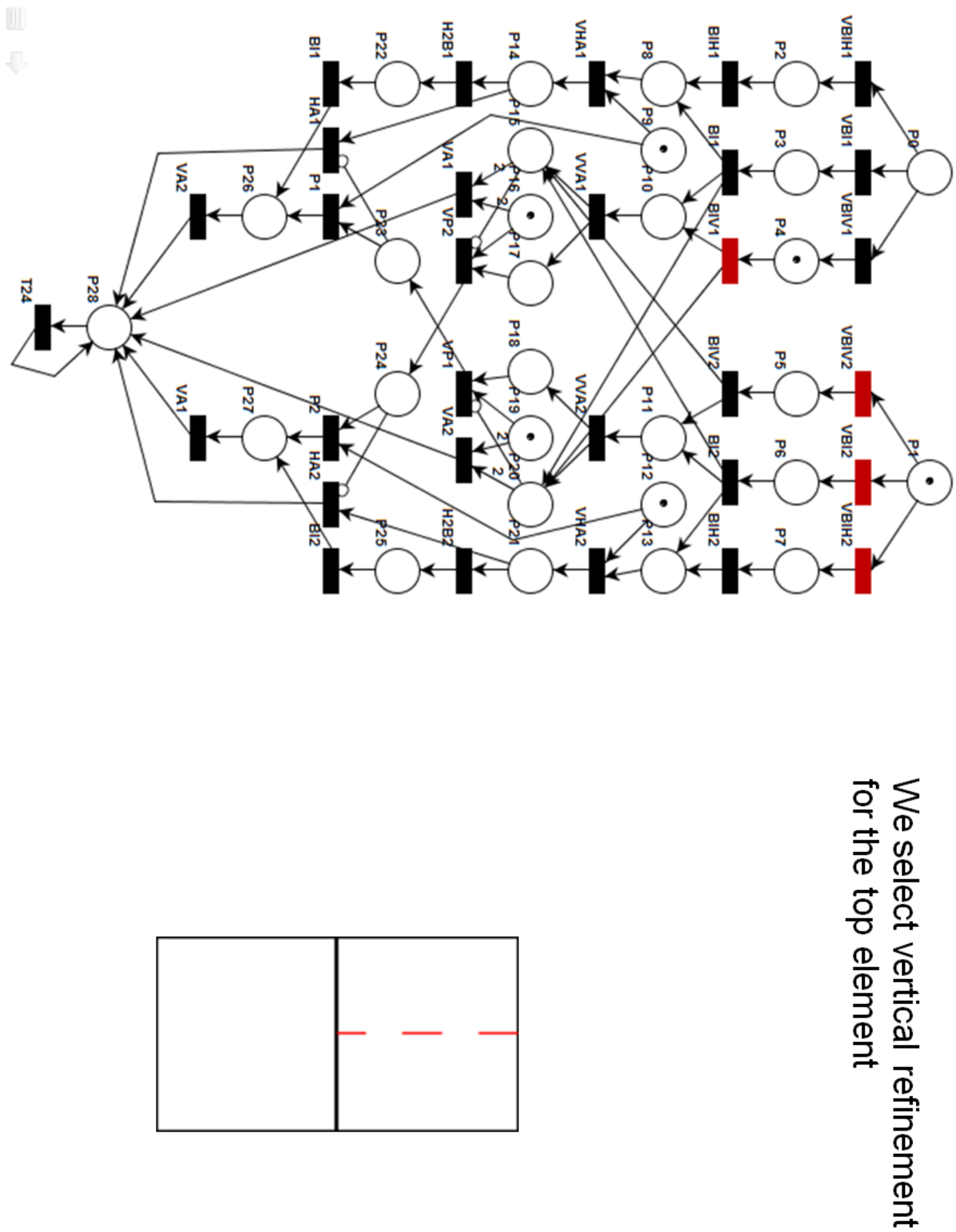


Figure A.11: Deadlock-free case. Stage 1

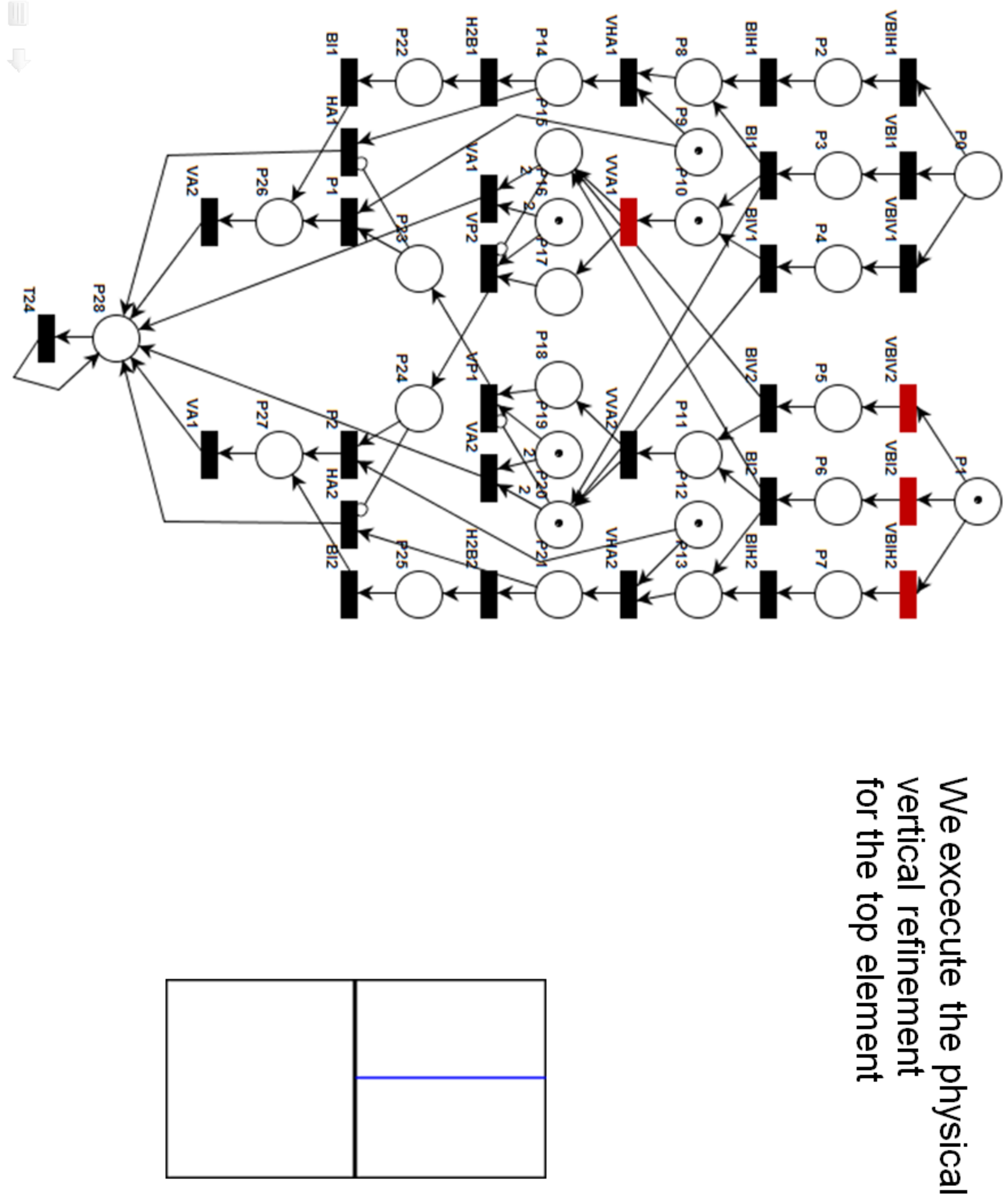


Figure A.12: Deadlock-free case. Stage 2

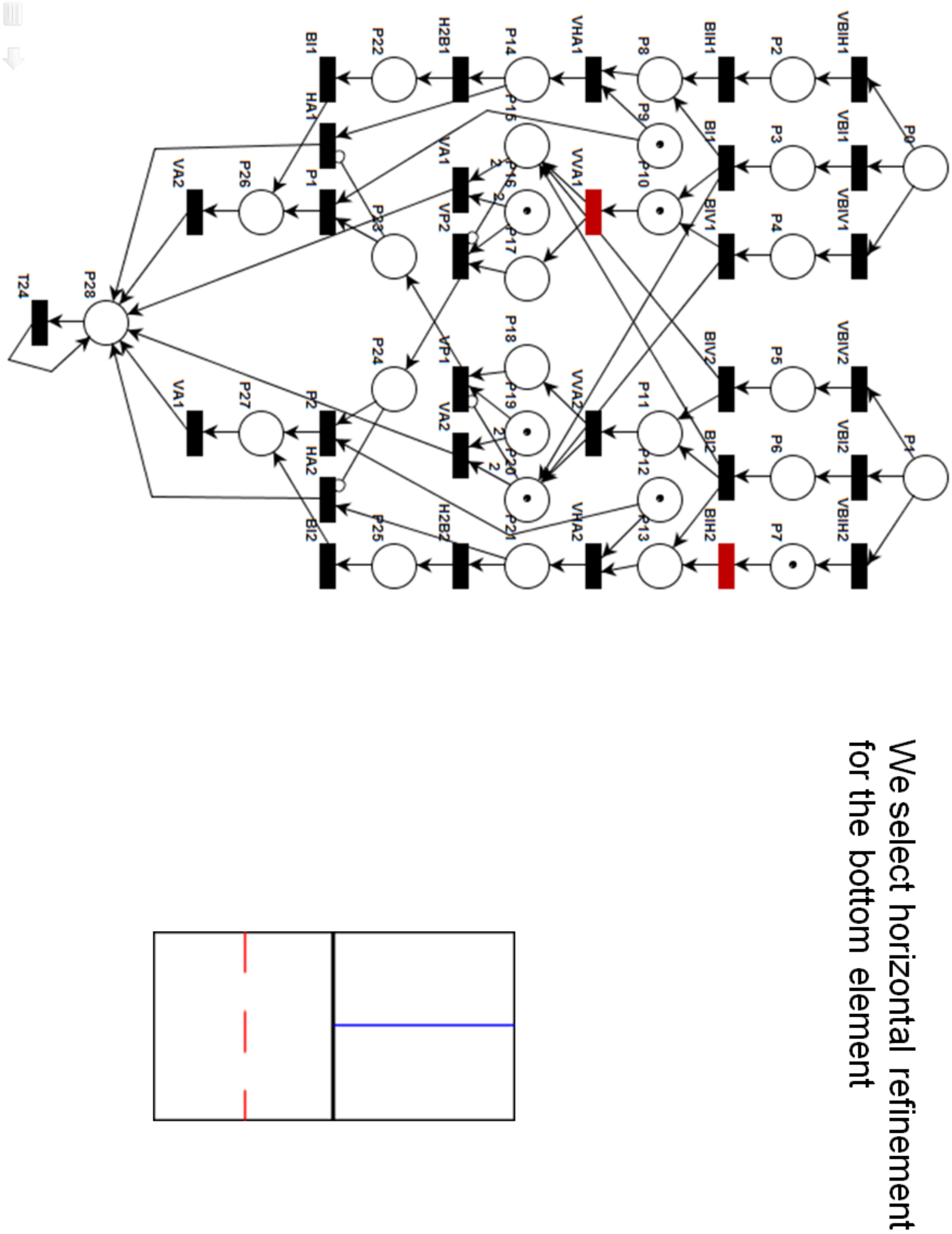


Figure A.13: Deadlock-free case. Stage 3

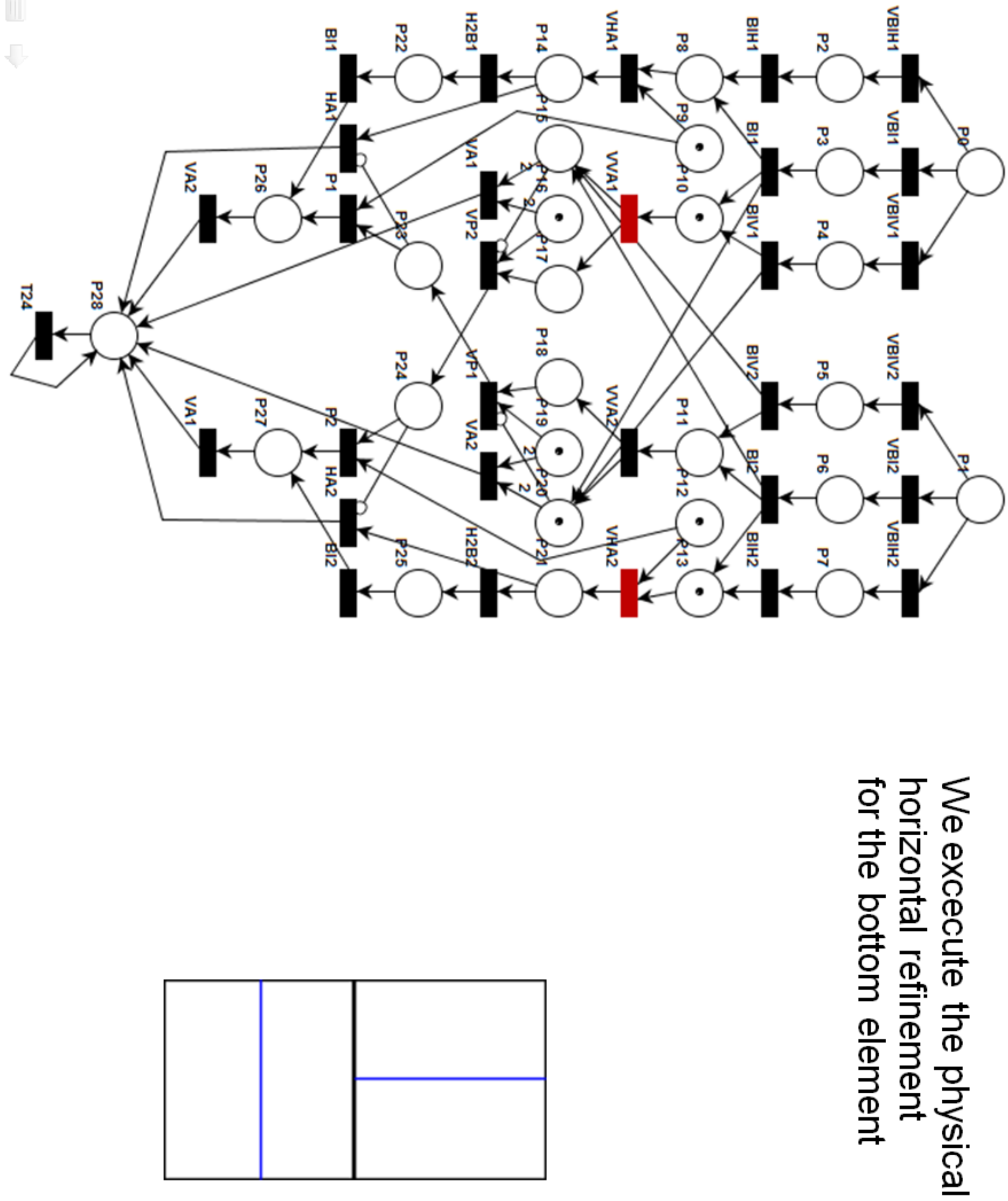


Figure A.14: Deadlock-free case. Stage 4

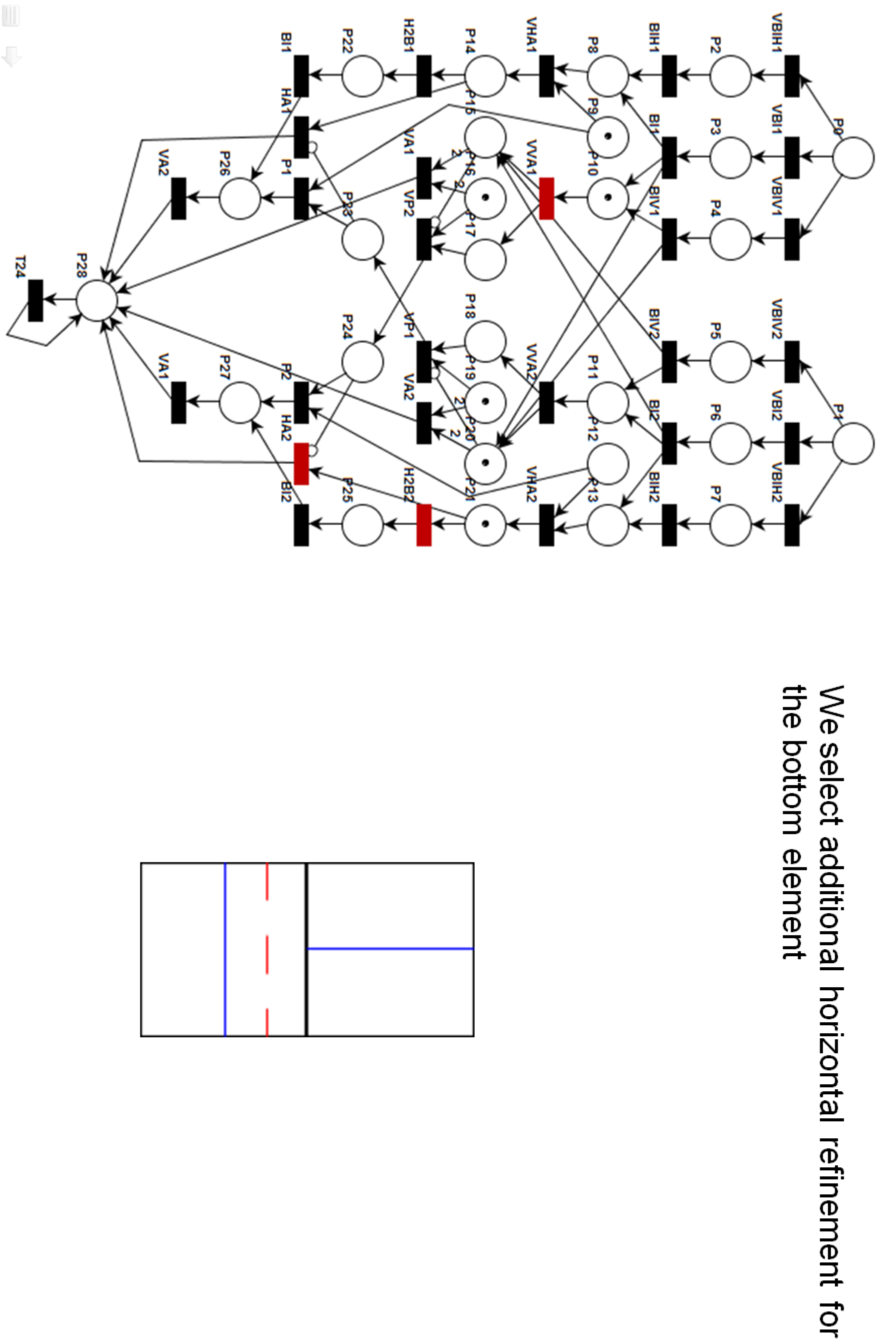


Figure A.15: Deadlock-free case. Stage 5

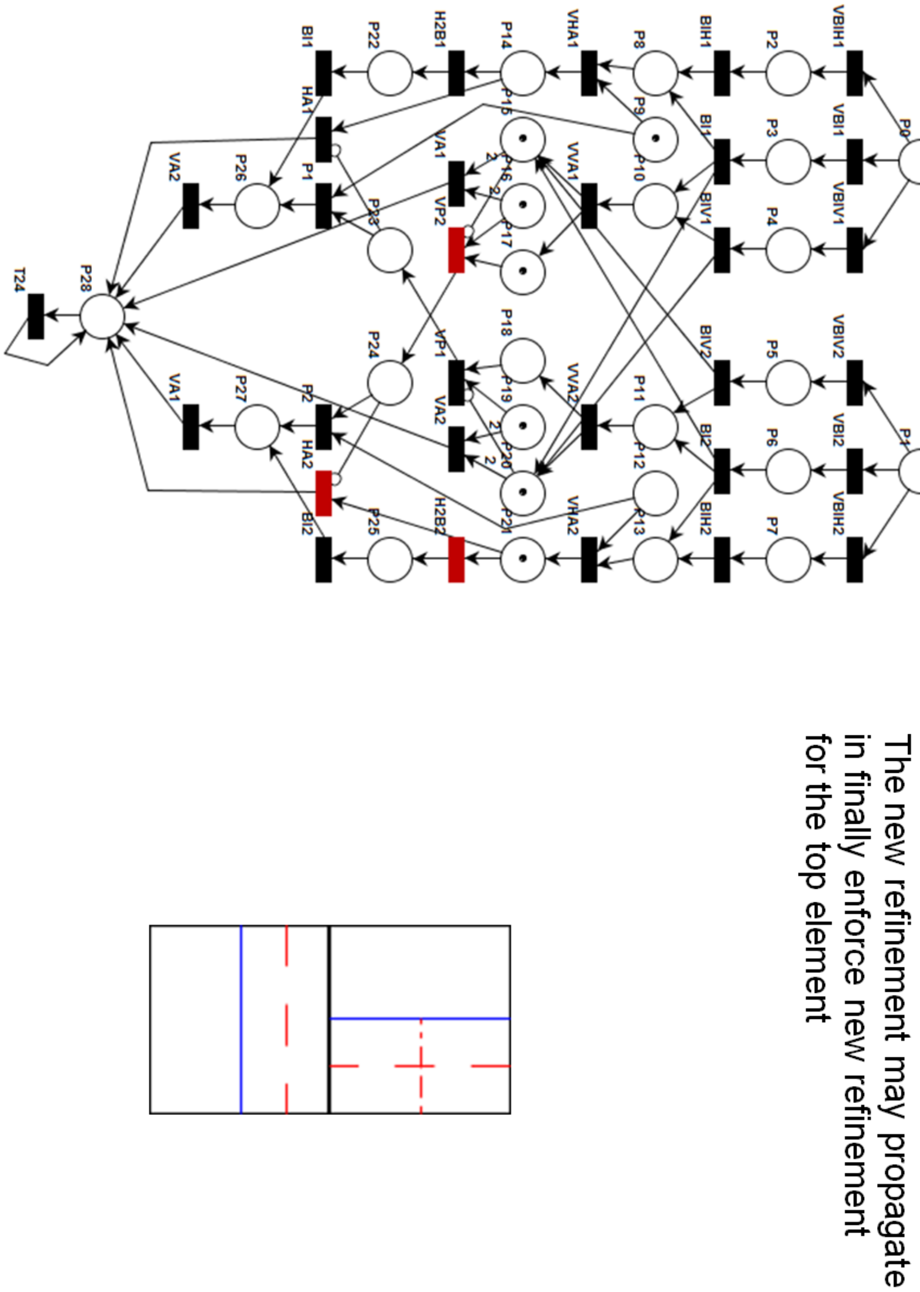


Figure A.16: Deadlock-free case. Stage 6

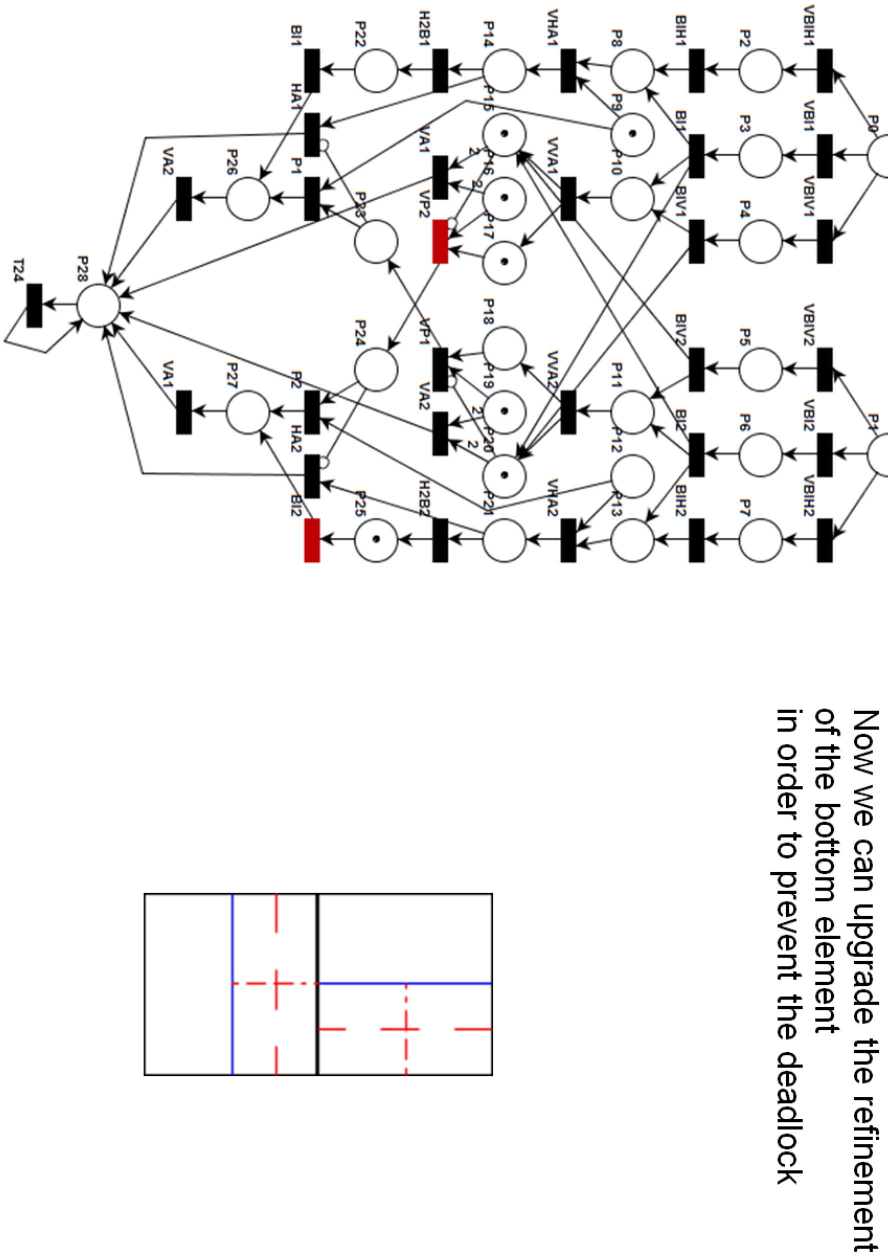


Figure A.17: Deadlock-free case. Stage 7

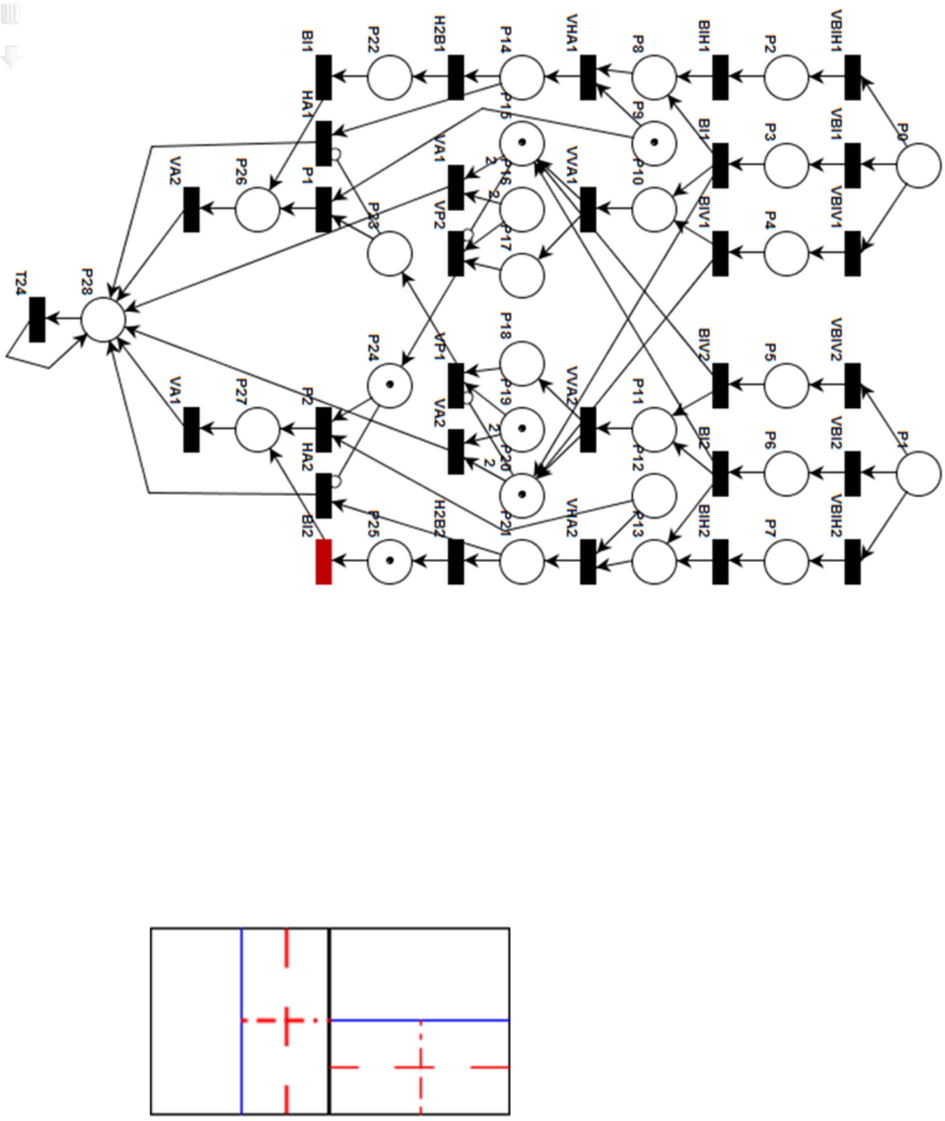


Figure A.18: Deadlock-free case. Stage 8

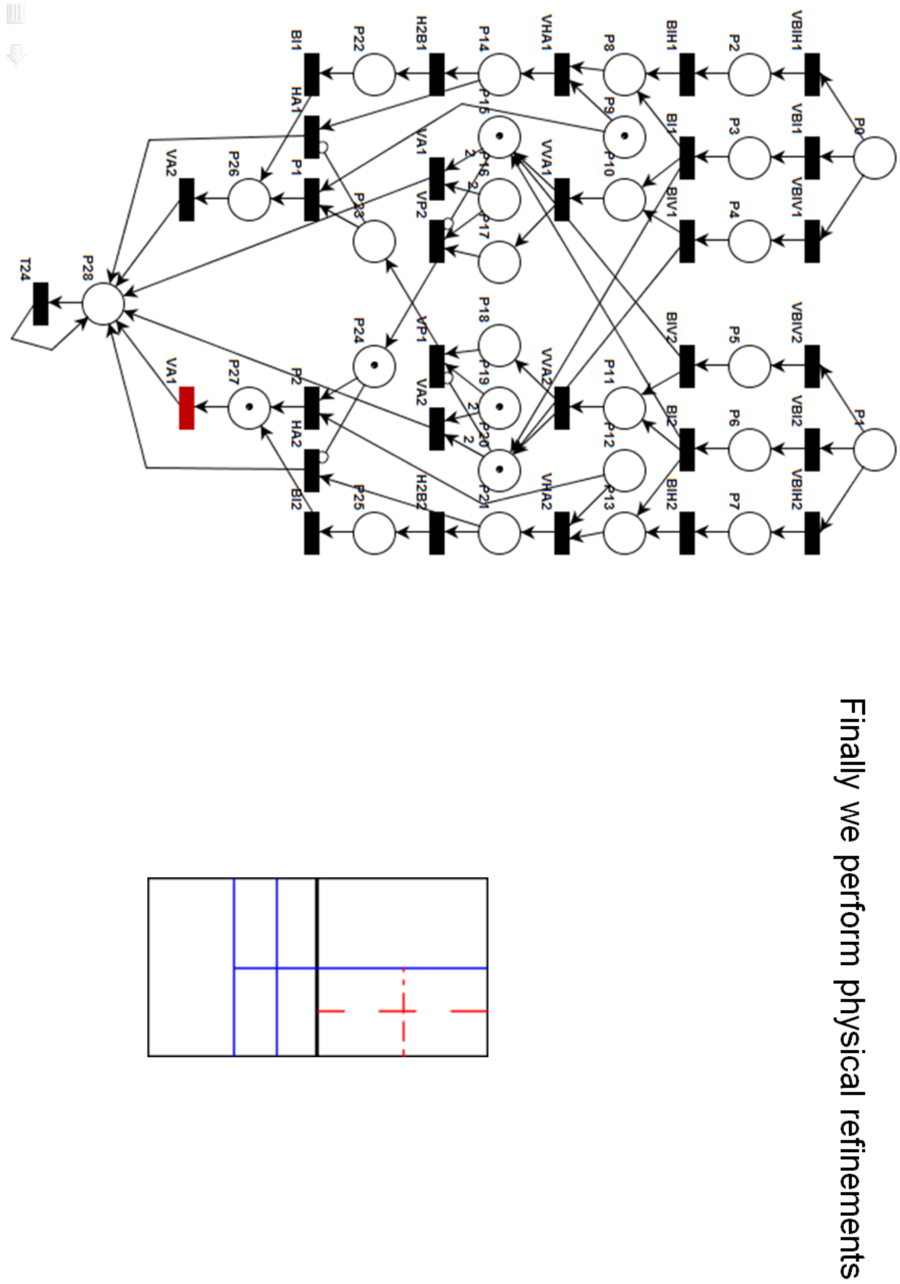


Figure A.19: Deadlock-free case. Stage 9

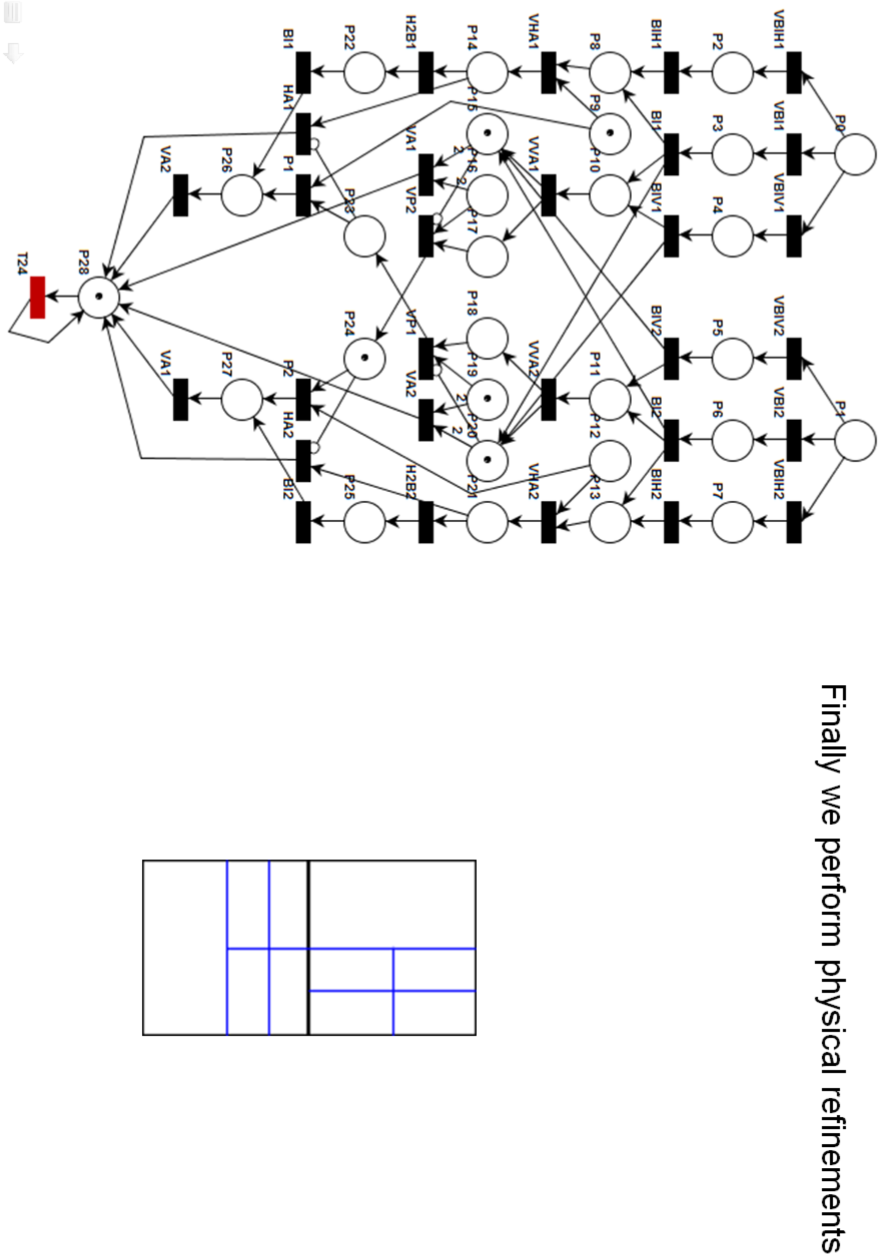


Figure A.20: Deadlock-free case. Stage 10

Appendix B

Introduction to hp adaptive Finite Element Method

This appendix introduces the hp-adaptive Finite Element Method (hp-FEM), the numerical method for solving Partial Differential Equations (PDE) in variational form. The method starts with either two-dimensional [10] or three-dimensional [11] boundary-value elliptic PDE transformed into so-called weak (variational) formulation of the form (B.1): Find $u \in V$ such that

$$b\left(u,v\right) = l\left(v\right) \quad \forall v \in V \tag{B.1}$$

where $b\left(u,v\right)$ and $l\left(v\right)$ are some problem dependent bilinear and linear functionals, and

$$V = \{v : \int_{\Omega} \|v\|^2 + \|\nabla v\|^2 dx < \infty, tr\left(v\right) = 0 \text{ on } \Gamma_D\} \tag{B.2}$$

is the functional Sobolev space over an open set Ω called the domain, and Γ_D is the part of boundary of Ω where Dirichlet boundary conditions are defined.

For a given domain Ω the hp-FEM consists in constructing a finite-dimensional subspace $V_{hp} \subset V$ with a finite-dimensional polynomial basis $\{e^i_{hp}\}_{i=1,\ldots,N_{hp}}$. The subspace V_{hp} is constructed by partitioning the domain Ω into so-called finite elements. Considerations in this thesis are restricted to rectangular elements in two dimensions or hexahedral elements in three dimensions. The basis functions are constructed by gluing together the so-called shape functions constructed over vertices, edges, and interiors of finite elements in two dimensions, or over vertices, edges, faces, and interiors of finite elements in three dimensions.

B.1. Two dimensional rectangular finite element

Figure B.1 presents an exemplary two-dimensional mesh consisting of rectangular finite elements with vertices, edges and interiors. Figure B.2 presents shape functions defined over vertices, edges and interiors of rectangular finite elements of the mesh. Let us introduce four shape functions over the four vertices of the two dimensional rectangular element $\{(\xi_1, \xi_2) : \xi_i \in [0,1], i = 1,2\}$:

$$\begin{aligned}
\hat{\phi}_1(\xi_1,\xi_2) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\\
\hat{\phi}_2(\xi_1,\xi_2) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\\
\hat{\phi}_3(\xi_1,\xi_2) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\\
\hat{\phi}_4(\xi_1,\xi_2) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)
\end{aligned} \tag{B.3}$$

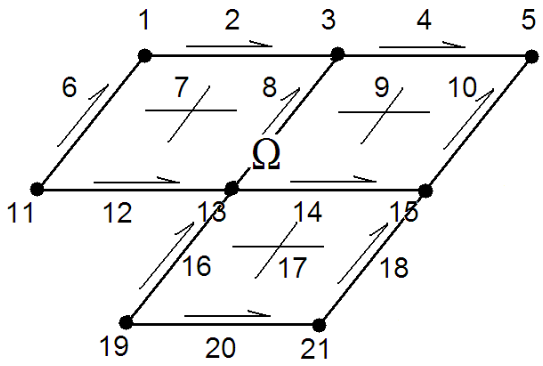


Figure B.1: A mesh with rectangular finite elements.

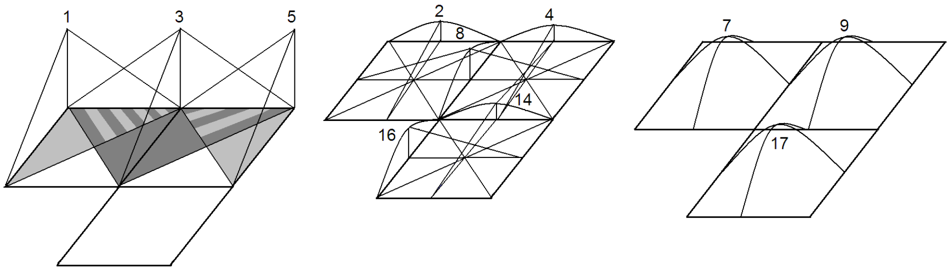


Figure B.2: Shape functions defined over vertices, edges and interiors of two-dimensional mesh with rectangular finite elements.

$p_i - 1$ shape functions over each of the four edges of the element

$$
\begin{aligned}
\hat{\phi}_{5,j}(\xi_1,\xi_2) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2) \quad j = 1,...,p_1 - 1\\
\hat{\phi}_{6,j}(\xi_1,\xi_2) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2) \quad j = 1,...,p_2 - 1\\
\hat{\phi}_{7,j}(\xi_1,\xi_2) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2) \quad j = 1,...,p_3 - 1\\
\hat{\phi}_{8,j}(\xi_1,\xi_2) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2) \quad j = 1,...,p_4 - 1
\end{aligned} \tag{B.4}
$$

where p_i is the polynomial order of approximation utilized over the i-th edge, and $(p_h - 1) \times (p_v - 1)$ shape functions over an element interior

$$\hat{\phi}_{9,ij}(\xi_1,\xi_2) = \hat{\chi}_{2+i}(\xi_1)\hat{\chi}_{2+j}(\xi_2) \quad i = 1,...,p_h - 1, j = 1,...,p_v - 1 \tag{B.5}$$

where (p_h, p_v) are the horizontal and vertical polynomial orders of approximation utilized over an element interior, and

$$
\begin{aligned}
\hat{\chi}_1(\xi) &= 1 - \xi\\
\hat{\chi}_2(\xi) &= \xi\\
\hat{\chi}_l(\xi) &= (1-\xi)\xi(2\xi - 1)^{l-3}, l = 4,...,p+1
\end{aligned} \tag{B.6}
$$

where p is the polynomial order of approximation over an edge. The above shape functions defined over element vertices, edges and interior are glued together for adjacent elements in order to form a global basis functions. These basis functions are utilized to approximate a solution of the weak form of the PDE being solved. For more details please refer to [10].

B.2. Three dimensional rectangular finite element

The three dimensional hexahedral finite element $\{(\xi_1, \xi_2, \xi_3) : \xi_i \in [0, 1], i = 1, 3\}$ is defined with eight shape functions over the eight vertices of the element:

$$\begin{aligned}
\hat{\phi}_1(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3) \\
\hat{\phi}_2(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3) \\
\hat{\phi}_3(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3) \\
\hat{\phi}_4(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3) \\
\hat{\phi}_5(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3) \\
\hat{\phi}_6(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3) \\
\hat{\phi}_7(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3) \\
\hat{\phi}_8(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3)
\end{aligned} \tag{B.7}$$

$p_i - 1$ shape functions over each of the twelve edges of the element

$$\begin{aligned}
\hat{\phi}_{9,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3) \quad j = 1, ..., p_1 - 1 \\
\hat{\phi}_{10,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_1(\xi_3) \quad j = 1, ..., p_2 - 1 \\
\hat{\phi}_{11,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3) \quad j = 1, ..., p_3 - 1 \\
\hat{\phi}_{12,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_1(\xi_3) \quad j = 1, ..., p_4 - 1 \\
\hat{\phi}_{13,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3) \quad j = 1, ..., p_5 - 1 \\
\hat{\phi}_{14,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_2(\xi_3) \quad j = 1, ..., p_6 - 1 \\
\hat{\phi}_{15,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3) \quad j = 1, ..., p_7 - 1 \\
\hat{\phi}_{16,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_2(\xi_3) \quad j = 1, ..., p_8 - 1 \\
\hat{\phi}_{17,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j = 1, ..., p_9 - 1 \\
\hat{\phi}_{18,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j = 1, ..., p_{10} - 1 \\
\hat{\phi}_{19,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j = 1, ..., p_{11} - 1 \\
\hat{\phi}_{20,j}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j = 1, ..., p_{12} - 1
\end{aligned} \tag{B.8}$$

where p_i is the polynomial order of approximation utilized over the i-th edge. Let us also define $(p_{ih}-1)\times(p_{iv}-1)$ shape functions over each of six faces of the finite element

$$\begin{aligned}
\hat{\phi}_{21}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_{2+k}(\xi_2)\hat{\chi}_1(\xi_3) \quad j = 1, ..., p_{13h} - 1, \quad k = 1, ..., p_{13v} - 1 \\
\hat{\phi}_{22}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_{2+k}(\xi_2)\hat{\chi}_2(\xi_3) \quad j = 1, ..., p_{14h} - 1, \quad k = 1, ..., p_{14v} - 1 \\
\hat{\phi}_{23}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+k}(\xi_3) \quad j = 1, ..., p_{15h} - 1, \quad k = 1, ..., p_{15v} - 1 \\
\hat{\phi}_{24}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_{2+k}(\xi_3) \quad j = 1, ..., p_{16h} - 1, \quad k = 1, ..., p_{16v} - 1 \\
\hat{\phi}_{25}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+k}(\xi_3) \quad j = 1, ..., p_{17h} - 1, \quad k = 1, ..., p_{17v} - 1 \\
\hat{\phi}_{26}(\xi_1, \xi_2, \xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_{2+k}(\xi_3) \quad j = 1, ..., p_{18h} - 1, \quad k = 1, ..., p_{18v} - 1
\end{aligned} \tag{B.9}$$

where p_{ih}, p_{iv} are the polynomial orders of approximations in two directions in the i-th face local coordinates system. Finally, let us define $(p_x - 1) \times (p_y - 1) \times (p_z - 1)$ basis functions over an element interior

$$\hat{\phi}_{27,ij}(\xi_1, \xi_2) = \hat{\chi}_{2+i}(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_{2+k}(\xi_3) \quad i = 1, ..., p_x - 1, j = 1, ..., p_y - 1, k = 1, ..., p_z - 1 \tag{B.10}$$

where (p_x, p_y, p_z) are the polynomial orders of approximation in three directions, respectively, utilized over an element interior. The above shape functions defined over element vertices, edges, faces, and interior are glued together for adjacent elements in order to form a global basis functions. These basis functions are utilized to approximate a solution of the weak form of the PDE being solved. For more details please refer to [11].

Appendix C

Basic Definitions of Petri nets

All definitions in this Appendix are taken from [58].

C.1. Simple Petri net

Definition C.1.1. *A Petri net is a 5-tuple* $N = (P, T, F, W, M_0)$ *where:*

- P *and* T *are disjoint finite sets of* places *and* transitions*, respectively;*
- $F \subset (P \times T) \cup (T \times P)$ *is a set of* arcs *(or flow relations);*
- $W : F \longrightarrow Z$ *is an arc multiset, so that the count (or* weight*) for each arc is a measure of the arc multiplicity;*
- $M_0 : P \longrightarrow Z$ *is a place multiset, where* Z *is a countable set. It is commonly described with reference to Petri net diagrams as* initial marking.

The *preset* of a transition t is the set of its input places: ${}^{\bullet}t = \{s \in S \mid W(s,t) > 0\}$; its *postset* is the set of its output places: $t^{\bullet} = \{s \in S \mid W(t,s) > 0\}$. Definitions of pre- and postsets of places are analogous.

Firing a transition t in a marking M consumes $W(s,t)$ tokens from each of its input places s, and produces $W(t,s)$ tokens in each of its output places s.

A transition is *enabled* (it may fire) in M if there are enough tokens in its input places for the consumptions to be possible, i.e. iff $\forall s : M(s) \geq W(s,t)$.

C.2. Hierarchical Petri net

For a variable v, $\theta(v)$ denotes *type* of v, i. e. the set of all possible values of v.

For an expression x, $\Omega(x)$ denotes the set of all variables in x, and $\theta(x)$ denotes *type* of x, i. e. the set of all possible values that can result from evaluation of x.

For a set of variables V, the *type* of V is defined as $\theta(V) = \{\theta(v) : v \in V\}$.

Bool denotes the set of boolean values ($\{false, true\}$).

For an arc a, $P(a)$ denotes the *place* node of a, and $T(a)$ denotes the *transition* node of a.

Definition C.2.1. *Colored Petri net is a tuple $N = (\Sigma, P, T, A, \gamma, C, G, E, M_0)$ where:*

- Σ *is a non-empty finite set of* types *(*colors*), whereof each is a non-empty set;*
- P *is a non-empty finite set of* places*;*
- T *is a non-empty finite set of* transitions*;*
- A *is a non-empty finite set of* arcs*, and* $P \cap T = P \cap A = T \cap A = \emptyset$*;*
- $\gamma : A \longrightarrow (P \times T) \cup (T \times P)$ *is a function associating each arc with an ordered pair of nodes (places and transitions);*
- $C : P \longrightarrow \Sigma$ *is a function associating each place with a set of token types (colors) allowed for given place;*
- G *is a function associating each transition with a* guard *expression such that* $\forall t \in T : \theta(G(t)) \subseteq Bool \wedge \theta(\Omega(G(t))) \subseteq \Sigma$*;*
- E *is a function associating each arc with its* weight *such that* $\forall a \in A : \theta(E(a)) \subseteq 2^{C(P(a))^*} \wedge \theta(\Omega(E(a))) \subseteq \Sigma$*;*
- M_0 *is an* initial marking *such that* $\forall p \in P : M_0(p) \in 2^{C(p)^*}$.

For a transition t, $In(t)$ denotes a set of places *input* to t and $Out(t)$ denotes a set of places *output* to t.

Transition substitution is a method of constructing a hierarchical Petri net by substituting a transition with another (flat or hierarchical) Petri net called *sub-page*.

Places in the *super-page* connected to the substituted transition with arcs are called *sockets*. For given substituted transition t, place p is called *input socket* if $p \in In(t)$, or p is called *output socket* if $p \in Out(t)$.

Sub-page places bound to *super-page sockets* are called *ports*. Each *port* may have one of the following types: *input port* (In), *output port* (Out), *input-output port* (I/O) or *general port* (Gen).

Place fusion is a set of indistinguishible places, i. e. all places belonging to the same fusion must have the same type and the same marking at any state of the Petri net. There are three types of *place fusions*:

- *Global fusion* (FG) - places belonging to such *fusion* may come from different *pages*;
- *Page fusion* (FP) - places belonging to such *fusion* may come from different *instances* of the same *page* in the entire Petri net;
- *Instance fusion* (FI) - places belonging to such *fusion* must come from the same *page instance*.

Definition C.2.2. *Hierarchical colored Petri net is a tuple $N = (S, SN, SA, PN, PT, PA, FS, FT, PP)$ where:*

- S *is a set of* pages*, whereof each is a colored non-hierarchical Petri net and elements of* P_S*,* T_S *and* A_S *for each pair of* pages *are disjoint;*
- $SN \subseteq T$ *is a set of* substituted transitions*;*
- SA *is a function associating each* substituted transition *with its* sub-page *such that no* page *is its own* sub-page*;*
- $PN \subseteq P$ *is a set of* ports*;*
- PT *is a function associating each* port *with its type (In, Out, I/O, Gen);*
- PA *is a function associating each* substituted transition *with a two-argument relation such that:*

  - sockets *are bound to* ports, *i. e.* $\forall t \in SN : PA(t) \subseteq (In(t) \cup Out(t)) \times PN_{SA(t)}$, *where* $PN_{SA(t)}$ *denotes* ports *on* t*'s immediate* sub-page*;*
  - *corresponding* sockets *and* ports *have the same type, while* general ports *(type Gen) can be bound to sockets of any type;*
  - *bound places (*sockets *and* ports*) have the same type and initial marking.*

- $FS \subseteq 2^P$ *is a finite set of* place fusions *such that places belonging to the same* fusion *must have the same type and initial marking;*
- FT *is a function associating each* place fusion *with its type* (FG, FP, FI)*;*
- PP *is a multiset of main pages.*

For additional definitions and explanations related to Petri nets please refer to [58].

List of figures

Bibliography

[1] Alvarez-Aramberria, J.—Pardo, D.—Barucq, H.: Inversion of Magnetotelluric Measurements using Multigoal Oriented hp-Adaptivity, Procedia Computer Science, Vol. 18, 2013, pp. 1564-1573.

[2] Banaś, K.—Michalik, K.: Design and development of an adaptive mesh manipulation module for detailed FEM simulation of flows. Procedia Computer Science, Vol. 1, 2010, No. 1, pp. 2043-2051.

[3] Bao, G.—Hu, G.—Liu, D.: An h-adaptive finite element solver for the calculations of the electronic structures. Journal of Computational Physics, Vol. 231, 2012, No.14, pp. 4967–4979.

[4] Babuška, I.—Rheinboldt, W.: Error Estimates for Adaptive Finite Element Computations. SIAM Journal of Numerical Analysis, Vol.15, 1978, No.4, pp. 736–754.

[5] Banaś K.:A Model for Parallel Adaptive Finite Element Software. Proceedings of 15th International Conference on Domain Decomposition Methods, Freie Universitat Berlin, 2003.

[6] Beal M. W., Shephard M. S., A General Topology-Based Mesh Data Structure, International Journal for Numerical Methods in Engineering, 40 (1997) 1573-1596

[7] Becker, R.—Kapp, H.—Rannacher, R.: Adaptive Finite Element Methods for Optimal Control of Partial Differential Equations: Basic Concept. SIAM Journal on Control and Optimisation, Vol. 39, 2000, No.1, pp. 113–132.

[8] Belytschko, T.—Tabbar, M.: H-Adaptive finite element methods for dynamic problems, with emphasis on localization. International Journal for Numerical Methods in Engineering, Vol. 36, 1993, No. 24, pp. 4245–4265.

[9] Chung, E.—Kimber, T.—Kirby, B.—Master, T.—Worthington, M.—Knottenbelt, W.: Petri nets group project final report, *http://pipe2.sourceforge.net/documents/PIPE2 Final Report.pdf*

[10] Demkowicz, L., 2006: Computing with hp-Adaptive Finite Elements, Vol. I. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications. Chapman and Hall/Crc Applied Mathematics and Nonlinear Science, 2006.

[11] Demkowicz, L.—Kurtz, J.—Pardo, D.—Paszyński M.—Rachowicz, W.—Zdunek A.: Computing with hp-Adaptive Finite Elements, Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications. Chapman and Hall/Crc Applied Mathematics and Nonlinear Science, 2007.

[12] Demkowicz, L.—Pardo, D.—Rachowicz, W.: 3D hp-Adaptive Finite Element Package (3Dhp90) Version 2.0. The Ultimate Data Structure for Three-Dimensional Anisotropic hp-Refinements. TICAM Report 02-24, 2002.

[13] Demkowicz, L.—Rachowicz, W.—Devloo, P.: A Fully Automatic hp-Adaptivity. Journal of Scientific Computing, Vol. 17, 2001, No. 1-3, pp. 127-155.

[14] Devine, K. D.—Flaherty, J. E.: Parallel Adaptive hp-Refinement Techniques for Conservation Laws. Applied Numerical Mathematics, Vol. 20, 1996, pp. 367-386.

[15] Edwards, H. C.: A Parallel Infrastructure for Scalable Adaptive Finite Element Methods and its application to Least Squares C-infinity Collocation. PhD. Dissertation, The University of Texas at Austin, 1997.

[16] Edwards, H. C.: SIERRA Framework Version 3: Core Services Theory and Design. SAND2002-3616, Albuquerque, NM: Sandia National Laboratories, 2002.

[17] Edwards ,H. C.—Stewart, J. R.: SIERRA, A Software Environment for Developing Complex Multiphysics Applications. Computational Fluid and Solid Mechanics, Proc. First MIT Conf. Cambridge MA, 2001.

[18] Edwards, H. C.—Stewart, J. R.—Zepper J. D.: Mathematical Abstractions of the SIERRA Computational Mechanics Framework. Proceedings of the Fifth World Congress on Computational Mechanics, Vienna Austria, 2002.

[19] Errikson, K.—Johnson, C.: Adaptive Finite Element Methods for Parabolic Problems I: A Linear Model Problem. SIAM Journal on Numerical Analysis, Vol. 28, 1991, No.1, pp. 43–77.

[20] Flasiński M., Schaefer R., Quasi context sensitive graph grammars as a formal model of FE mesh generation, Computer-Assisted Mechanics and Engineering Science, 3 (1996) 191-203

[21] Gawad J., Paszyński M., Matuszyk P., Madej L., Cellular automata coupled with hp-adaptive Finite Element Method applied to simulation of austenite-ferrite phase transformation with a moving interface, Steel Research, ISSN 1611-3683, 79 (2008) 579-586.

[22] Gomez-Revuelto, I.—Garcia-Castillo, L. E.—Llorente-Romano, S.,—Pardo, D.: A three-dimensional self-adaptive hp finite element method for the characterization of waveguide discontinuities. Computer Methods in Applied Mechanics and Engineering, Vol. 249-252, 2012, pp.62-74.

[23] Gurgul, P.—Sieniek, M.—Paszyński, M.—Madej, L.—Collier, N.: Two-dimensional HP-adaptive Algorithm for Continuous Approximations of Material Data Using Space Projection. Computer Science, Vol. 14, 2013, No.1.,pp.97–112.

[24] Grabska E., Theoretical Concepts of Graphical Modeling. Part One: Realization of CP-Graphs. Machine Graphics and Vision 2, 1 (1993) 3-38

[25] Grabska E., Theoretical Concepts of Graphical Modeling. Part Two: CP-Graph Grammars and Languages. Machine Graphics and Vision 2, 2 (1993) 149-178

[26] Grabska E., Graphs and Designing, Graph Transformation in Computer Science, H.J. Schneider and H.Ehrig (Eds.), Lecture Notes in Computer Science, Springer-Verlag, 776, 1994.

[27] Grabska E, Hliniak G., Structural Aspects of CP-Graph Languages. Schedae Informaticae 5 (1993) 81-100

[28] Habel A, Hyperedge Replacement: Grammars and Languages, Lectures Notes in Computer Science, 643, Springer, 1992.

[29] Kardani, M.—Nazem, M.—Abbo, A.—Sheng, D.—Sloan, S.: Refined h-adaptive finite element procedure for large deformation geotechnical problems. Computational Mechanics, Vol.49, 2012, No.1, pp.21–33.

[30] Kyoungjoo K.: Finite Element Modeling of Electromagnetic Radiation and Induced Heat Transfer in Human Body. PhD Thesis, The University of Texas at Austin, 2013.

[31] Laszloffy, A.—Long, J.—Patra, A. K.: Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive hp finite element simulations. Parallel Computing, Vol. 26, 2000, pp.1765-1788.

[32] Nam, M.J.—Pardo, D.—Torres-Verdin, C. :Simulation of borehole-eccentered triaxial induction measurements using a Fourier hp finite element method. Geophysics, Vol. 78, 2013, No. 2, pp. D41-D52.

[33] Niemi, A.—Babuška, I.—Pitkäranta, J.—Demkowicz, L.: Finite element analysis of the Girkmann problem using the modern hp-version and the classical h-version. Engineering with Computers, Vol. 28, 2012, No.2, pp.123–134.

[34] Nochetto, R. H.—Siebert, K. G.—Veeser, A.: Multiscale, Nonlinear and Adaptive Approximation. Springer, 2009, pp. 409–542.

[35] Pardo, D.—Torres-Verdin, C.: Sensitivity analysis for the appraisal of hydrofractures in horizontal wells with borehole resistivity measurements. Geophysics, Vol. 78, 2013, No. 4, pp. D209-D222.

[36] Pardo D., Demkowicz L., Torres-Verdin C., Paszyński M., A Goal Oriented hp-Adaptive Finite Element Strategy with Electromagnetic Applications. Part II: Electrodynamics. Computer Methods in Applied Mechanics and Engineering, special issue in honor of Prof. Ivo Babuška, 196 (2007) 3585-3597.

[37] Pardo D., Torres-Verdin C., Paszyński M., Simulation of 3D DC Borehole Resistivity Measurements with a Goal-Oriented hp Finite Element Method. Part II: Through-Casing Resistivity Instruments, Computational Geophysics, 12 (2008) 83-89.

[38] Pardo, D.—Demkowicz, L.—Torres-Verdín, C.—Paszynski, M.: A self-adaptive goal-oriented hp-finite element method with electromagnetic applications. Part II: Electrodynamics. Computer Methods in Applied Mechanics and Engineering, Vol. 196, 2007, No. 37, pp. 3585–3597.

[39] Pardo, D.—Demkowicz, L.—Torres-Verdín, C.—Paszynski, M.:Two-Dimensional High-Accuracy Simulation of Resistivity Logging-While-Drilling (LWD) Measurements Using a Self-Adaptive Goal-Oriented hp Finite Element Method. SIAM Journal on Applied Mathematics, Vol. 66, 2006, No.6, pp. 2085–2106.

[40] Pardo, D.—Torres-Verdín, C.—Paszynski, M.: Simulations of 3D DC borehole resistivity measurements with a goal-oriented hp finite-element method. Part II: through-casing resistivity instruments. Computational Geosciences, Vol. 12, 2008, No. 1, pp.83–89.

[41] Paszynska, A.—Grabska, E.—Paszynski, M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part I. Fundamenta Informaticae, Vol. 114, 2012, No. 2, pp.149–182.

[42] Paszynska, A.—Grabska, E.—Paszynski, M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part II. Fundamenta Informaticae, Vol. 114, 2012, No. 2, pp.183–201.

[43] Paszynska, A.—Paszynski, M.—Grabska, E.: Graph Transformations for Modeling hp-Adaptive Finite Element Method with Mixed Triangular and Rectangular Elements. Lecture Notes in Computer Science, Vol. 5545, 2009, pp.875-884.

[44] Paszynska, A.—Paszynski, M.—Grabska, E.: Graph Transformations for Modeling hp-Adaptive Finite Element Method with Triangular Elements. Lecture Notes in Computer Science, Vol. 5103, 2008, pp.604–613.

[45] Paszynska, A.—Paszynski, M.—Szymczak, A.—Pardo, D.: Petri Nets for Detecting a 3D Deadlock Problem in Hp-adaptive Finite Element Simulations. Procedia Computer Science, Vol. 9, 2012, pp. 1434-1443.

[46] Paszyński M., Demkowicz L., Pardo D., Verification of Goal-Oriented hp-Adaptivity, Computers and Mathematics with Applications, 50, 8-9 (2005) 1395-1404.

[47] Paszyński, M.—Demkowicz, L.: Parallel Fully Automatic hp-Adaptive 3D Finite Element Package. Engineering with Computers, Vol. 22, 2006, pp.255-276.

[48] Paszyński M., Kurtz J., Demkowicz L.,: Parallel Fully Automatic hp-Adaptive 2D Finite Element Package, Computer Methods in Applied Mechanics and Engineering, 195, 7-8 (2006) 711-741.

[49] Paszyński M., Maciol P.: Application of the Fully Automatic 3D hp Adaptive Code to Orthotropic Heat Transfer in Structurally Graded Materials, Journal of Material Processing Technology, 177, 1-3 (2006) 68-71.

[50] Paszyński M., Romkes A., Collister E., Meiring J., Demkowicz L., Willson, C. G.: On the Modeling of Step-and-Flash Imprint Lithography using Molecular Statics Models, ICES Report 05-38 (2005).

[51] Patra A. K.: Parallel HP Adaptive Finite Element Analysis for Viscous Incompressible Fluid Problems. PhD. Dissertation, University of Texas at Austin, 1999.

[52] Patro, S. K.—Selvam, P. R.—Bosch, H.: Adaptive h-finite element modeling of wind flow around bridges. Engineering Structures, Vol. 48, 2013, pp. 569–577.

[53] Plazek, J.: Implementation Issues of Computational Fluid Dynamics Algorithms on Parallel Computers. Lecture Notes in Computer Science, Vol. 1697, 1999, pp.349-355.

[54] Plazek, J.: Scalable CFD Computations Using Message-Passing and Distributed Shared Memory Algorithms. Lecture Notes in Computer Science, Vol. 1908, 2000, pp.282-288.

[55] Plazek, J.–Banaś, K.—Kitowski, J.: Comparison of Message Passing and Shared Memory Implementations of the GMRES method on MIMD computers, Scientific Programming, Vol. 9, 2001, pp.195-209.

[56] Remacle, J. F.—Xiangrong, L.—Shephard, M.S.—Flaherty J.E.: Anisotropic Adaptive Simulations of Transient Flows using Discontinuous Galerkin Methods. International Journal of Numerical Methods in Engineering, Vol. 00, 2000, pp. 1-6

[57] Spicher A., Michel O., Giavitto J.: Declarative Mesh Subdivision Using Topological Rewriting in MGS, International Conference on Graph Transformation, Enschede, The Netherlands, September 2010, Lecture Notes in Computer Science 6372 (2010) 298-313.

[58] Szpyrka, M.: Petri nets for modeling and analysis of concurrent systems, Wydawnictwa Naukowo-Techniczne, Warsaw, Poland, 2008.

[59] Szymczak, A.—Paszynska, A.—Paszynski, M.—Pardo, D.: Anisotropic 2D mesh adaptation in hp-adaptive Finite Element Method. Procedia Computer Science, Vol. 4, 2011, pp.1818–1827.

[60] Szymczak, A.—Paszynska, A.—Paszynski, M.—Pardo, D.: Preventing deadlock during anisotropic 2D mesh adaptation in hp-adaptive FEM. Journal of Computational Science, Vol. 4, 2013, No. 3, pp.170–179.

[61] Szymczak A., Paszyński M.: Grach grammar based model of concurrency for self-adaptive hp Finite Element Method, Lecture Notes in Computer Science, vol. 6067, 2010, 95-104.

[62] Szymczak, A.—Paszynski, M.—Pardo, D., A.—Paszynska: Petri Nets modeling of dead-end refinement problems in a 3D anisotropic hp-adaptive finite element method, accepted to Computing and Informatics, 2014.

[63] Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services, *http://www.cs.sandia.gov/zoltan/*

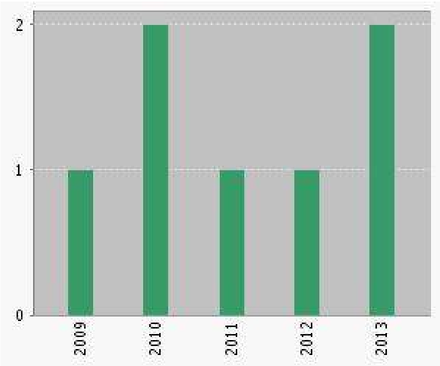
Citation Report: 7

(from All Databases)

You searched for: **AUTHOR:** (Szymczak A*) *AND* **ADDRESS:** (Poland) **...More**

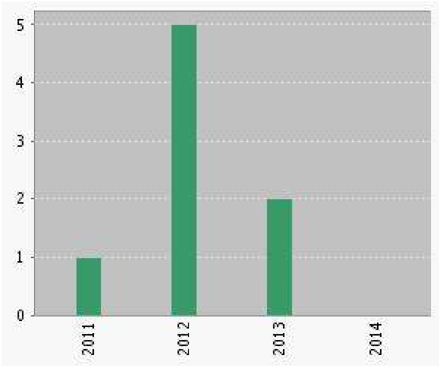
This report reflects citations to source items indexed within All Databases.

Published Items in Each Year



The latest 20 years are displayed.

Citations in Each Year



The latest 20 years are displayed.

Results found: 7

Sum of the Times Cited [?] : 8

Sum of Times Cited without self-citations [?] : 6

Citing Articles [?] : 6

Citing Articles without self-citations [?] : 5

Average Citations per Item [?] : 1.14

h-index [?] : 2

Sort by: Times Cited -- highest to lowest | Page 1 of 1

	2010	2011	2012	2013	2014	Total	Average Citations per Year
Use the checkboxes to remove individual items from this Citation Report or restrict to items published between 1900 and 2014 Go	0	1	5	2	0	8	2.00
1. **International Conference on Computational Science, ICCS 2011 Anisotropic 2D mesh adaptation in hp-adaptive FEM** By: Szymczak, Arkadiusz; Paszynska, Anna; Paszynski, Maciej; et al. Edited by: Sato, M; Matsuoka, S; Sloot, PMA; et al. Conference: International Conference on Computational Science (ICCS) on the Ascent of Computational Excellence Location: Campus Nanyang Technolog Univ, Singapore, SINGAPORE Date: 2011 Sponsor(s): Elsevier; Univ Tsukuba, Ctr Computat Sci PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE (ICCS) Book Series: Procedia Computer Science Volume: 4 Pages: 1818-1827 Published: 2011	0	0	2	1	0	3	0.75
2. **Graph grammar-based multi-thread multi-frontal parallel solver with trace theory-based scheduler** By: Obrok, Pawel; Pierzchala, Pawel; Szymczak, Arkadiusz; et al. Edited by: Sloot, PMA; Albada, GDV; Dongarra, J Book Group Author(s): ICCS Conference: International Conference on Computational Science (ICCS) Location: Univ Amsterdam, Amsterdam, NETHERLANDS Date: MAY 31-JUN 02, 2010 Sponsor(s): NWO, Netherlands Org Sci Res; KNAW, Royal Netherlands Acad Arts & Sci; Elsevier; Univ Amsterdam ICCS 2010 - INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE, PROCEEDINGS Book Series: Procedia Computer Science Volume: 1 Issue: 1 Pages: 1987-1995 Published: 2010	0	1	1	1	0	3	0.60
3. **Preventing deadlock during anisotropic 2D mesh adaptation in hp-adaptive FEM** By: Szymczak, Arkadiusz; Paszynska, Anna; Paszynski, Maciej; et al. JOURNAL OF COMPUTATIONAL SCIENCE Volume: 4 Issue: 3 Special Issue: SI Pages: 170-179 Published: MAY 2013	0	0	1	0	0	1	0.50

4.	**Graph Grammar Based Petri Nets Model of Concurrency for Self-adaptive hp-Finite Element Method with Rectangular Elements** By: Szymczak, Arkadiusz; Paszynski, Maciej Edited by: Wyrzykowski, R; Dongarra, J; Karczewski, K; et al. Conference: 8th International Conference on Parallel Processing and Applied Mathematics Location: Wroclaw, POLAND Date: SEP 13-16, 2009 Sponsor(s): Intel Corp; Hewlett-Packard Co; Microsoft Corp; IBM Corp; Action S A; AMD PARALLEL PROCESSING AND APPLIED MATHEMATICS, PT I Book Series: Lecture Notes in Computer Science Volume: 6067 Pages: 95-104 Part: I Published: 2010	0	0	1	0	0	1	0.20
5.	**Graph Grammar Based Direct Solver for hp-adaptive Finite Element Method with Point Singularities** By: Szymczak, Arkadiusz; Paszynska, Anna; Gurgul, Piotr; et al. Edited by: Alexandrov, V; Lees, M; Krzhizhanovskaya, V; et al. Conference: 13th Annual International Conference on Computational Science (ICCS) Location: Barcelona, SPAIN Date: JUN 05-07, 2013 Sponsor(s): Univ Amsterdam; Univ Tennessee; Nanyang Technol Univ; Barcelona Super Comp Ctr 2013 INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE Book Series: Procedia Computer Science Volume: 18 Pages: 1594-1603 Published: 2013	0	0	0	0	0	0	0.00
6.	**Petri nets for detecting a 3D deadlock problem in hp-adaptive finite element simulations** By: Paszynska, Anna; Paszynski, Maciej; Szymczak, Arkadiusz; et al. Edited by: Ali, H; Shi, Y; Khazanchi, D; et al. Conference: International Conference on Computational Science (ICCS) Location: Omaha, NE Date: JUN 04-06, 2012 Sponsor(s): Univ Nebraska; Universiteit Amsterdam; Univ Tennessee; Nanyang Technol Univ; Chinese Acad Sci Res Ctr Fictitious Econ & Data Sci; Elsevier; Univ Nebraska, Coll Informat Sci & Technol; Gallup Org; Union Pacific Railroad; Interpubl Grp Co (IPG) PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE, ICCS 2012 Book Series: Procedia Computer Science Volume: 9 Pages: 1434-1443 Published: 2012	0	0	0	0	0	0	0.00
7.	**Graph Grammar Based Petri Nets Model of Concurrency for Self-adaptive hp-Finite Element Method with Triangular Elements** By: Szymczak, Arkadiusz; Paszynski, Maciej Edited by: Allen, G; Seidel, E; Dongarra, J; et al. Conference: 9th International Conference on Computational Science Location: Baton Rouge, LA Date: MAY 25-27, 2009 Sponsor(s): Louisiana Univ, Ctr Computat & Technol; Univ Amsterdam; Univ Tennessee COMPUTATIONAL SCIENCE - ICCS 2009 Book Series: Lecture Notes in Computer Science Volume: 5545 Pages: 845-854 Published: 2009	0	0	0	0	0	0	0.00

Select Page   Save to Text File

Sort by: Times Cited -- highest to lowest   Page 1 of 1

Approximately 7 records matched your query of the 135,725,211 (contains duplicates) in the data limits you selected.