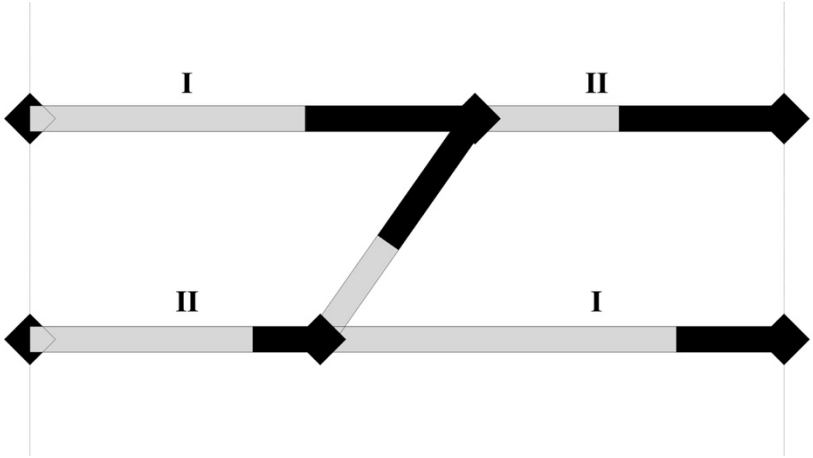
Perez Ramirez – Sferza – Alsayed Ahmad

26-05-16

PROPOSED ASSUMPTION FOR THE ORDER OF DEFORMATION PROBLEM

1. WHEN DO THE CURRENT ASSUMPTIONS FAIL?

Let us consider the most simple structure containing a connection and in particular the order of deformation depicted.

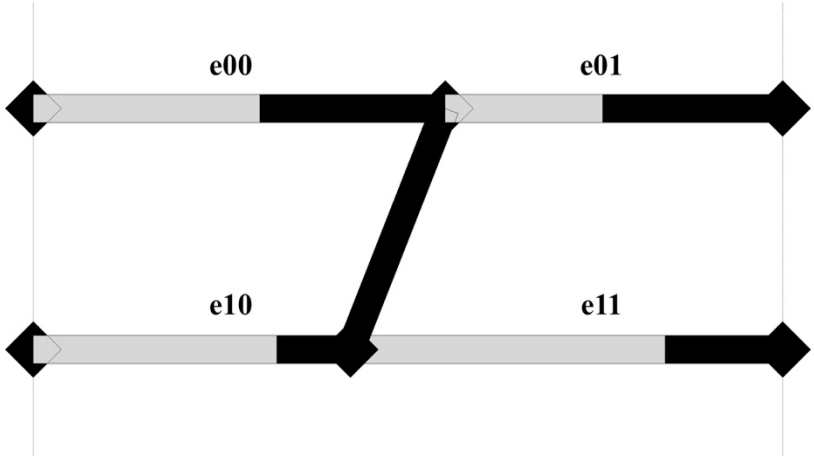


This possible solution comes directly from combinatorics: it's one out of the 4 possibilities.

In order to understand, if this solution is meaningful or should be discarded, we have to test it, deformation step after deformation step.

A. First deformation step

i. We start deforming as much as we can the components marked with I (i.e. e00 and e11).



ii. The connection deforms completely and prevents a further deformation of the other components.

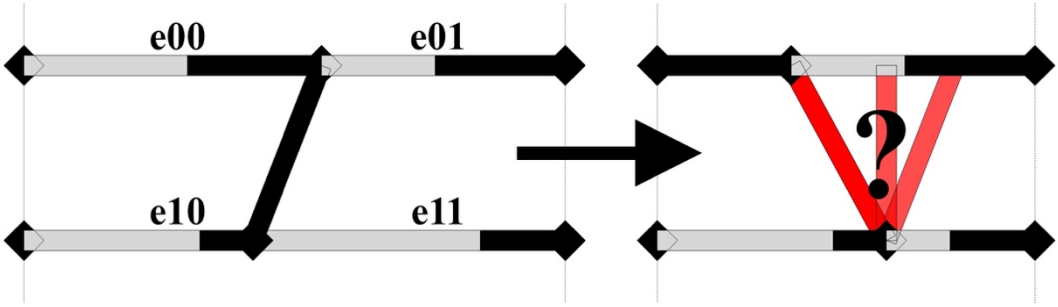
B. The next deformation step

i. Here comes the problem: the next deformation step isn't defined, if we bear in mind the following assumptions:

- *only one component per loadpath can deform at the same time,*
- *weaker components deform first,*
- *a component cannot deform partially and then re-start deforming,*
- *a connection cannot stretch.*

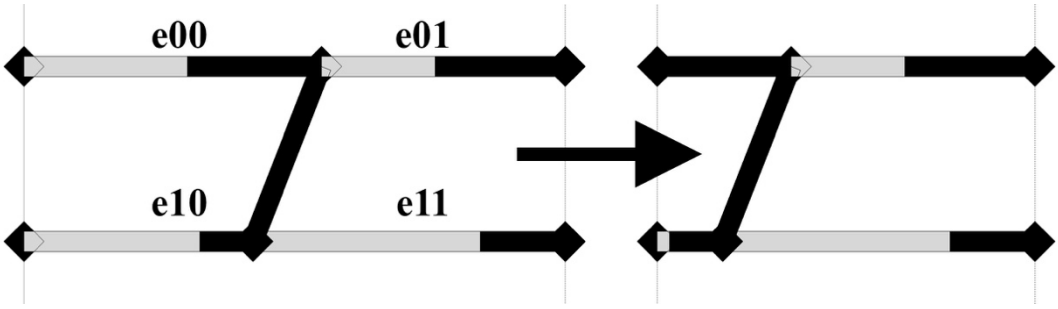
ii. Analysis of the possible next deformation steps:

Case 1: deformation of e00 and e11



The connection should deform even more and then elongate if needed, but this doesn't make sense, since the connection has become rigid after the first deformation step.

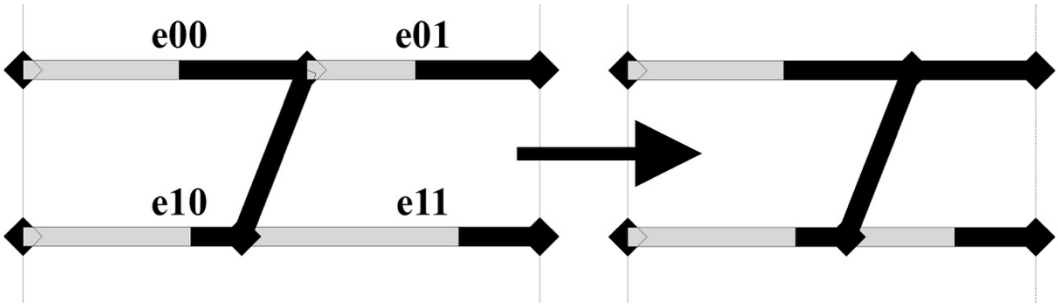
Case 2: deformation of e00 and e10



e10 deforms instead of e11, this doesn't make sense, since e11 is weaker than e10.

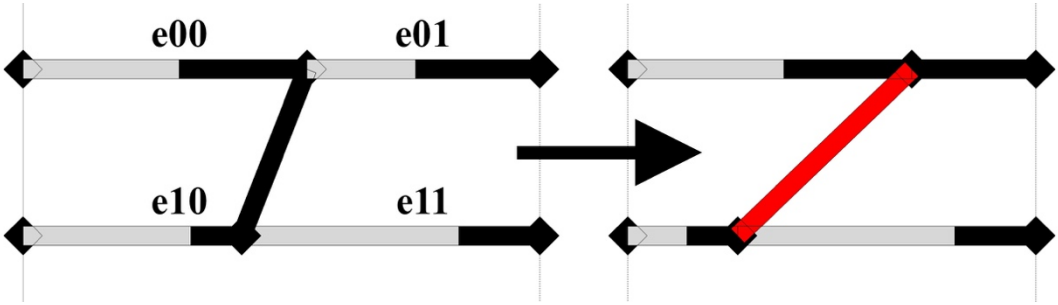
Furthermore, since the deformation of e11 doesn't continue after the first deformation step, e11 will remain undeformed at the end of the process, according to the 3rd assumption, and the structure depicted on the right will be considered completely deformed.

Case 3: deformation of e01 and e11



Same as case 2.

Case 4: deformation of e10 and e01



The connection should stretch, against the 4th assumption.

iii. Considering that all the possible next deformation step are not satisfactory, we've come to a deadlock.

2. A POSSIBLE SOLUTION

A. How to overcome the deadlock

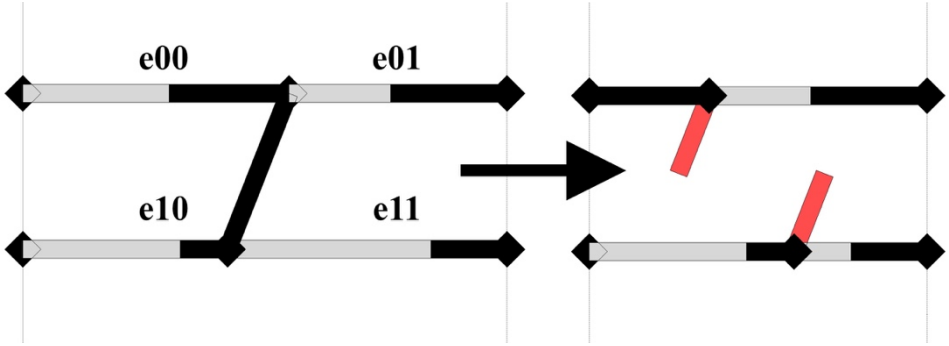
A good idea to go beyond the deadlock might be to ignore the presence of the connection as in case 1.

Another idea might be to work around the 3rd assumption and allow components to re-deform after a deadlock.

B. Two practical ways

i. The connection breaks

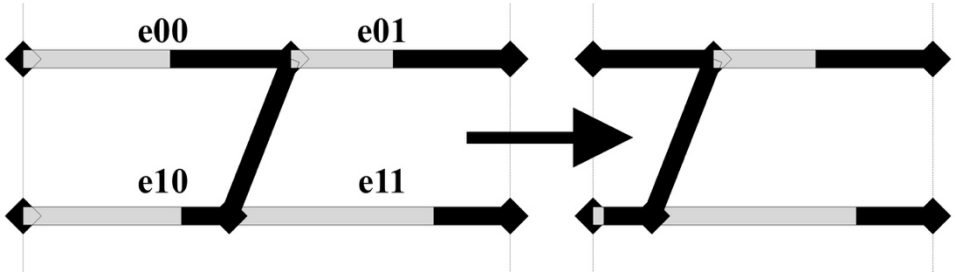
If the connection breaks, e00 and e11 will simply keep on deforming



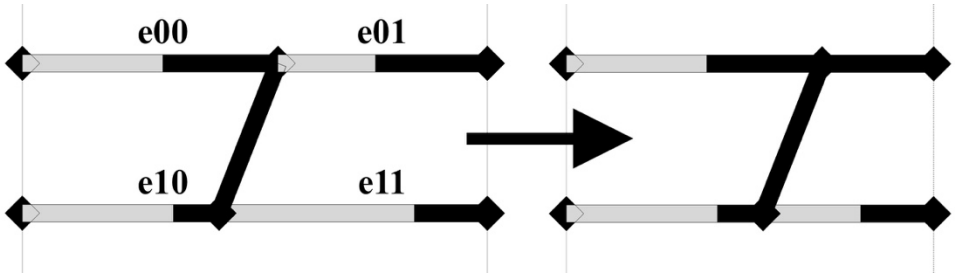
ii. The connection doesn't break

If the connection doesn't break, the deadlock will be considered as the initial configuration of a new problem with 2 possible solutions:

- *either e00 and e10 deform first*



- *or e01 and e11 deform first*



3. PRACTICAL IMPLEMENTATION

A. How to discriminate between breakable and unbreakable connections?

i. Deformable ratio criterion

One way to discriminate between breakable and unbreakable connections might be considering:

- *breakable those with a deformable ratio of 1*
- *unbreakable all the others*

This appears to be reasonable to us, because completely deformable components, have no length after a complete deformation and therefore are probably used to model the weakest components.

ii. Other criteria

Other criteria may be found in the future and one can classify them as:

- *Criteria based on the characteristics of the connection itself (as the proposed "Deformable-ratio criterion")*
- *Criteria based on the characteristics of the structure as a whole (e.g. disposition of masses, characteristics of neighbour components, …)*
- *Criteria based on the particular deformation history*

B. Using a flag to store the breakable/unbreakable information

i. Convenient for us

The easiest way to keep track of this information is adding an attribute to the Connection class in our code. The data to store is a simple boolean: self.breakable = True/False

ii. Versatile for you

The value of the flag will be:

- *directly read from the .xml, if you want to explicitly define it*
- *computed based on the deformable ratio criterion, if not specified in the .xml*
- *computed based on an other criterion: in case in the future you come up with a particular sophisticated criterion, it will be easy to extend our code and adapt its mode of operation to you needs*