# Erratum for *Constraint Processing*, Decther, R., Morgan Kaufmann Publishers, 2003.

## Course Notes for IAIP, Fall 2005

Rune Møller Jensen, Tarik Hadzic, and Peter Tiedemann

Department of Computer Science, IT University of Copenhagen,
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
{rmj,tarik,petert}@itu.dk

## 1 List of errors and inaccuracies

This errata list covers chapter 1 to 6 of the book which were used in the course Efficient AI Programming (IAIP) taught at the IT University of Copenhagen in the fall 2005.

### 1.1 page 16, join (inaccuracy)

The operative definition of join is imprecise. It is not fully clear that for every tuple in $R_S$ <u>all</u> matching tuples in $R_S$ are included in $R_S \bowtie R_T$.

### 1.2 page 19, o-notation (error)

The definition of $o(g(n))$ at the bottom of the page 19 is imprecise. It lacks the existential quantifier in the definition. So, instead of

$$o(g(n)) = \{f(n) : \text{for any positive constant } c \text{ and } n_0, \\ 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$$

it should say:

$$o(g(n)) = \{f(n) : \text{for any positive constant } c \ \underline{\text{there exists}}\ n_0, \\ 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$$

### 1.3 page 33 (typo)

The columns are indexed by $X_i$'s domains and the rows by $X_j$'s domain.

it should say:

The <u>rows</u> are indexed by $X_i$'s domains and the <u>columns</u> by $X_j$'s domain.

### 1.4 page 36 (typo)

$$\text{... two values } \{\mathit{true}, \mathit{false}\} \text{ or } \{0, 1\}.$$

it should say:

$$\text{... two values } \{\mathit{true}, \mathit{false}\} \text{ or } \{1, 0\}.$$

### 1.5 page 38 (error)

$$\text{... } M_1 \rightarrow (A \wedge C \rightarrow X).$$

it should say:

$$\text{... } M_1 \rightarrow (A \wedge C \leftrightarrow X).$$

### 1.6 page 44, Figure 2.11 (error)

$$D_1 = \{1, 3\}$$

it should say:

$$D_1 = \{2, 3\}$$

### 1.7 page 58, Figure 3.6 (typo)

$$\text{... AC-2 is applied.}$$

it should say:

$$\text{... AC-3 is applied.}$$

### 1.8 page 60, AC-4 algorithm (error)

Line 10 of AC-4 algorithm presented in Figure 3.7 allows selection of tuple $(x_j, a_j) \in S_{(x_i,a_i)}$ even though it has been previously selected and added to list $M$. Therefore, instead:

$$10 : \text{for each } (x_j, a_j) \text{ in } S_{(x_i,a_i)}$$

it should say:

$$10 : \text{for each } (x_j, a_j) \text{ in } S_{(x_i,a_i)} \text{ and } (x_j, a_j) \notin M$$

### 1.9 page 88, Proposition 4.1 (inaccuracy)

It is clear that a tree has induced width 1. However, that a graph with induced width 1 is a tree seems to imply an unusual definition of trees that does not require the graph to be connected.

### 1.10 page 101, Theorem 4.4 (typo)

$$(j < i)$$

it should say:

$$(j \leq i)$$

### 1.11 page 108, (typo)

> ... The induced width along $d_1$ and $d_2$ are 3 and 2, respectively...

it should say:

> ... The induced width along $d_2$ and $d_1$ are 3 and 2, respectively...

### 1.12 page 108, Figure 4.17 (error)

This figure seems to wrongly eliminate $C$ before $D$. Furthermore, there are errors in the tables.

### 1.13 page 125, Figure 4.18 (inaccuracy)

"The constraints can be organized to permit finding a tuple of a given constraint in worst-case logarithmic time:...". But on page 133 it can be done in $O(1)$. What is the difference in the underlying data structure?

### 1.14 page 133, procedure Select-Value-Forward-Checking (error)

In the test "**If** $D'_k$ is empty" the domains are reset but the algorithm does not break out of the for loop to try another element in $D'_i$. This must be done to ensure correctness of the approach.

### 1.15 page 139, procedure DVFC (error)

Remove the first line "$i \leftarrow i + 1$ (step forward to $x_S$)".

### 1.16 page 158, Definition 6.7 (inaccuracy)

Notice that the ancestors of $x$ on page 86 section 4.1 were defined as the <u>parents</u> of $x$.

### 1.17 page 160, Definition 6.8 (inaccuracy)

For the Graph-Based-Backjumping procedure to make sense (e.g., instantiating $l_i$ to $anc(x_i)$) and for Theorem 6.1 on page 161 to be correct, the relevant dead-end set for internal dead-ends must also include the ancestor set of the internal dead-end node it-self.