

The Set of Pareto-Nash Equilibria in Multicriteria Strategic Games

Victoria Lozan, Valeriu Ungureanu

Abstract

The paper investigates the notion of Pareto-Nash equilibrium as continuation of the works [2–4]. Problems and basic theoretical results are exposed. Method of intersection of graphs of best response mappings [3] is applied to solve the dyadic two-criteria games.

Mathematics Subject Classification 2000: 91A05, 91A06, 91A10, 91A43, 91A44.

Keywords: Noncooperative game, graph of best response mapping, intersection, multicriteria game, Pareto-Nash equilibrium, set of Pareto-Nash equilibria, dyadic game with mixed strategies.

1 Introduction

Consider the noncooperative strategic form game:

$$\Gamma = \langle \mathbf{N}, \{\mathbf{X}_{\mathbf{p}}\}_{p \in \mathbf{N}}, \{f_p^i(\mathbf{x})\}_{i=1}^{m_p}, p \in \mathbf{N} \rangle,$$

where

- $\mathbf{N} = \{1, 2, \dots, n\}$ is a set of players;
- $\mathbf{X}_{\mathbf{p}} \in \mathbf{R}^{k_p}$ is a set of strategies of player $p \in \mathbf{N}$;
- $k_p < +\infty, p \in \mathbf{N}$;
- and $\{f_p^i(\mathbf{x})\}_{i=1}^{m_p}$ are the p^{th} player cost functions defined on the Cartesian product $\mathbf{X} = \times_{p \in \mathbf{N}} \mathbf{X}_{\mathbf{p}}$.

Remark that each player has to solve singly the multi-criteria parametric optimization problem, where the parameters are strategic choices of the others players.

To exclude uncertainty, the well known definitions 1–3 and the corresponding notations are presented.

Definition 1. Strategy x_p' is "better" than x_p'' , if

$$\{f_p^i(x_p', x_{-p})\}_{i=1}^{m_p} \geq \{f_p^i(x_p'', x_{-p})\}_{i=1}^{m_p}, \forall x_{-p} \in \mathbf{X}_{-p},$$

and there exist at least one index $j \in \{1, \dots, m_p\}$ and a joint strategy $x_{-p} \in \mathbf{X}_{-p}$ for which

$$f_p^j(x_p', x_{-p}) > f_p^j(x_p'', x_{-p});$$

the last relationship is denoted as $x_p' \succeq x_p''$.

Player problem. The player p selects from his set of strategies the strategy $x_p^* \in \mathbf{X}_p$, $p \in \mathbf{N}$, for which all of his cost functions $\{f_p^i(x_p, x_{-p}^*)\}_{i=1}^{m_p}$ reach maximum values.

2 Pareto optimality

Definition 2. Strategy x_p^* is named effective (optimal in the sense of Pareto), if there does not exist other strategy $x_p \in \mathbf{X}_p$ so that $x_p \succeq x_p^*$.

Let us denote the set of effective strategies (solutions) of the player p by $\mathbf{ef} \mathbf{X}_p$. Any two effective strategies are equivalent or incomparable.

Theorem 1. If the sets $\mathbf{X}_p \in \mathbf{R}^{k_p}$, $p = \overline{1, n}$, are compact and the cost functions are continuous ($f_p^i(\mathbf{x}) \in C(\mathbf{X}_p)$, $i = \overline{1, m_p}$, $p = \overline{1, n}$), then the sets $\mathbf{ef} \mathbf{X}_p$, $p = \overline{1, n}$, are non empty ($\mathbf{ef} \mathbf{X} \neq \emptyset$).

The proof follows from the known results [1, 2].

Definition 3. Every element $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*) \in \mathbf{ef} \mathbf{X} = \times_{p \in \mathbf{N}} \mathbf{ef} \mathbf{X}_p$ is named effective or Pareto outcome (situation).

3 Synthesis Function

Solution of multi-criteria problem may be found by applying synthesis function, which may be interpreted as unique cost function of the player p ($p = \overline{1, n}$):

$$\begin{aligned} F_p(x) &= \sum_{i=\overline{1, m_p}} \lambda_i f_p^i(x_p, x_{-p}) \longrightarrow \max, \\ x_p &\in \mathbf{X}_p, \\ \sum_{i=\overline{1, m_p}} \lambda_i &= 1, \lambda_i \geq 0, i = \overline{1, m_p}. \end{aligned}$$

Theorem 2. *If x_p^* is a solution of mono-criterion problem*

$$F_p(x) = \sum_{i=\overline{1, m_p}} \lambda_i f_p^i(x_p, x_{-p}) \longrightarrow \max, x_p \in \mathbf{X}_p$$

with $\lambda_i > 0$, $i = \overline{1, m_p}$, $\sum_{i=\overline{1, m_p}} \lambda_i = 1$, then x_p^ is the efficient point for the given $x_{-p} \in \mathbf{X}_{-p}$.*

The theorem's proof follows from the sufficient Pareto condition with linear synthesis function [1, 2].

4 Pareto-Nash equilibriun

Consider the convex game Γ for which the sets of strategies are convex and the cost functions are concave in relation to respective player strategies, when the strategies of the other players are fixed.

Definition 4. *The point $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*) \in \mathbf{X}$ is a Pareto-Nash equilibrium, if and only if for any player p the relations*

$$F_p(x_p, x_{-p}^*) \leq F_p(x_p^*, x_{-p}^*) \equiv F_p(x^*), \forall x_p \in \mathbf{X}_p,$$

are verified.

As a corollary of the precedent two theorems follow.

Theorem 3. *If the sets \mathbf{X}_p , $p = \overline{1, n}$, of the convex game Γ are compact and the functions $\{f_p^i(\mathbf{x})\}_{i=1}^{m_p}$ are continuous on $\mathbf{X} = \times_{p \in N} \mathbf{X}_p$, then the convex game Γ has the Pareto-Nash equilibrium.*

Proof of the Theorem 3 follows from the known result [3].

The definition 4 may be formulated in other equivalent form:

Definition 5. *The point $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*) \in \mathbf{X}$ is a Pareto-Nash equilibrium, if and only if*

$$F(\mathbf{x}^*) = \left(\max_{x_1 \in \mathbf{X}_1} F_1(x_1, x_{-1}^*), \dots, \max_{x_n \in \mathbf{X}_n} F_n(x_n, x_{-n}^*) \right),$$

where $(x_p, x_{-p}^*) \equiv (x_1^*, x_2^*, \dots, x_{p-1}^*, x_p, x_{p+1}^*, \dots, x_n^*)$, $p = \overline{1, n}$.

So, the Pareto-Nash equilibrium requires from each player to choose his own strategy as the Pareto best response to the strategies chosen by other players.

Let us denote the graph of the mapping

$$\text{Argmax}_{x_p \in \mathbf{X}_p} F_p(x_p, x_{-p}) : \mathbf{X}_{-\mathbf{p}} \longrightarrow \mathbf{X}_p$$

by

$$Gr_p = \{(x_p, x_{-p}) \in \mathbf{X} : x_{-p} \in \mathbf{X}_{-\mathbf{p}}, x_p = \text{argmax}_{y_p \in \mathbf{X}_p} F_p(y_p, x_{-p})\}.$$

In such notation, by [4], the set of Pareto-Nash equilibrium is:

$$\mathbf{PNE} = \bigcap_{p=\overline{1, n}} Gr_p,$$

where $\mathbf{X}_{-\mathbf{p}} = \times_{i \in N \setminus \{p\}} \mathbf{X}_i$.

As an illustration of previous notions and method of PNE set determination, let us consider the following example.

Example 1. Consider the discrete game of two players. Each player has two strategies and two cost functions. The players have to maximize the values of both cost functions. The values of the cost functions are associated with the matrix elements:

$$A = \begin{pmatrix} 4, 3 & 7, 7 \\ 6, 6 & 8, 4 \end{pmatrix},$$

$$B = \begin{pmatrix} 5, -1 & 2, 4 \\ 4, 3 & 6, 2 \end{pmatrix}.$$

First of all the sets of effective strategies $\mathbf{ef} \mathbf{X}$ and $\mathbf{ef} \mathbf{Y}$ are determined. Elements of $\mathbf{ef} \mathbf{X}$ and $\mathbf{ef} \mathbf{Y}$ are included in angle brackets.

$$A = \begin{pmatrix} 4, 3 & \langle 7, 7 \rangle \\ 6, 6 & \langle 8, 4 \rangle \end{pmatrix}, B = \begin{pmatrix} 5, -1 & \langle 2, 4 \rangle \\ \langle 4, 3 \rangle & \langle 6, 2 \rangle \end{pmatrix}.$$

$$\mathbf{PNE} = \mathbf{ef} \mathbf{X} \cap \mathbf{ef} \mathbf{Y} = \{(1, 2), (2, 2)\}$$

with the costs $\{((7, 7), (2, 4)), ((8, 4), (6, 2))\}$.

Theorem 4. If the sets \mathbf{X}_p , $p = \overline{1, n}$, in the convex game Γ are compact and the functions $F_p(x)$ are continuous on $\mathbf{X} = \times_{p \in \mathbf{N}} \mathbf{X}_p$, then the convex game Γ has the Pareto-Nash equilibrium.

The proof follows from [4, 2, 5] and the theorems 1–3, also.

5 Dyadic two-criteria game with mixed strategies

Consider dyadic two-criteria game with mixed strategies. The sets of strategies are:

$$\mathbf{X} = \{(x_1, x_2) : x_1 + x_2 = 1, x_1 \geq 0, x_2 \geq 0\},$$

$$\mathbf{Y} = \{(y_1, y_2) : y_1 + y_2 = 1, y_1 \geq 0, y_2 \geq 0\}.$$

The cost functions are bilinear, i.e. the functions are linear for fixed opponent strategy:

$$f_1^1(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T A \mathbf{y},$$

$$f_1^2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T B \mathbf{y},$$

$$f_2^1(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T C \mathbf{y},$$

$$f_2^2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T D \mathbf{y},$$

where $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$, $A, B, C, D \in \mathbf{R}^{2 \times 2}$. For each player consider the synthesis function:

$$F_1(x) = \lambda_1 f_1^1(x) + \lambda_2 f_1^2(x) \longrightarrow \max,$$

$$F_2(x) = \mu_1 f_2^1(x) + \mu_2 f_2^2(x) \longrightarrow \max.$$

By applying substitutions: $\lambda_1 = \lambda > 0$, and $\lambda_2 = 1 - \lambda > 0$, $\mu_1 = \mu > 0$ and $\mu_2 = 1 - \mu > 0$, we obtain:

$$F_1(\mathbf{x}, \mathbf{y}) = \lambda f_1^1(\mathbf{x}, \mathbf{y}) + (1 - \lambda) f_1^2(\mathbf{x}, \mathbf{y}) = \lambda \mathbf{x}^T A \mathbf{y} + (1 - \lambda) \mathbf{x}^T B \mathbf{y},$$

$$F_2(\mathbf{x}, \mathbf{y}) = \mu f_2^1(\mathbf{x}, \mathbf{y}) + (1 - \mu) f_2^2(\mathbf{x}, \mathbf{y}) = \mu \mathbf{x}^T C \mathbf{y} + (1 - \mu) \mathbf{x}^T D \mathbf{y}.$$

By applying obvious transformations:

$$x_1 = x, x_2 = 1 - x, 1 \geq x \geq 0,$$

$$y_1 = y, y_2 = 1 - y, 1 \geq y \geq 0,$$

the second equivalent game is obtained:

$$F_1(x, y) = (\alpha(\lambda)y + \beta(\lambda))x + \alpha_0(\lambda)y + \beta_0(\lambda),$$

$$F_2(x, y) = (\gamma(\mu)x + \delta(\mu))y + \gamma_0(\mu)x + \delta_0(\mu),$$

$$x, y \in [0, 1], \lambda, \mu \in [0, 1],$$

where:

$$\alpha(\lambda) = (a_{11} - a_{12} - a_{21} + a_{22} - b_{11} + b_{12} + b_{21} - b_{22})\lambda + b_{11} - b_{12} - b_{21} + b_{22},$$

$$\beta(\lambda) = (a_{12} - a_{22} - b_{12} + b_{22})\lambda + b_{12} - b_{22},$$

$$\alpha_0(\lambda) = (a_{21} - a_{22} - b_{21} + b_{22})\lambda + b_{21} - b_{22},$$

$$\beta_0(\lambda) = (a_{22} - b_{22})\lambda + b_{22},$$

$$\gamma(\mu) = (c_{11} - c_{12} - c_{21} + c_{22} - d_{11} + d_{12} + d_{21} - d_{22})\mu + d_{11} - d_{12} - d_{21} + d_{22},$$

$$\delta(\mu) = (c_{21} - c_{22} - d_{21} + d_{22})\mu + d_{21} - d_{22},$$

$$\gamma_0(\mu) = (c_{12} - c_{22} - d_{12} + d_{22})\mu + d_{12} - d_{22},$$

$$\delta_0(\mu) = (c_{22} - d_{22})\mu + d_{22}.$$

The graphs of Pareto best response mappings are:

$$\mathbf{Gr}_1 = \begin{cases} (1, y), & \text{if } \alpha(\lambda)y + \beta(\lambda) > 0, \\ (0, y) & \text{if } \alpha(\lambda)y + \beta(\lambda) < 0, \\ [0, 1] \times y, & \text{if } \alpha(\lambda)y + \beta(\lambda) = 0, \end{cases}$$

$$\mathbf{Gr}_2 = \begin{cases} (x, 1), & \text{if } \gamma(\mu)x + \delta(\mu) > 0, \\ (x, 0), & \text{if } \gamma(\mu)x + \delta(\mu) < 0, \\ x \times [0, 1], & \text{if } \gamma(\mu)x + \delta(\mu) = 0. \end{cases}$$

The solutions of equations $\alpha(\lambda)y + \beta(\lambda) = 0$ and $\gamma(\mu)x + \delta(\mu) = 0$ are $y(\lambda) = -\frac{\beta(\lambda)}{\alpha(\lambda)}$ and $x(\mu) = -\frac{\delta(\mu)}{\gamma(\mu)}$. Vertical asymptotes of the respective hyperboles are determined by relations $\alpha(\lambda) = 0$ and $\gamma(\mu) = 0$ and they are denoted by λ_α and μ_γ , respectively.

If the solution λ_α does not belong to the interval $(0, 1)$, then y belongs to the interval with extremities $[y(0), y(1)]$. If the extremity value is negative, it is replaced by 0, if it is greater than 1, it is replaced by 1.

If λ_α belongs to the interval $(0, 1)$, then the graph \mathbf{Gr}_1 will be represented by two rectangles and one edge ($[(0, 0), (0, 1)]$ or $[(1, 0), (1, 1)]$) of the square $[0, 1] \times [0, 1]$ or two edges and one rectangle of the square $[0, 1] \times [0, 1]$. Other graphs are possible in the case when λ_α does not belong to the interval $(0, 1)$ and they are described below.

Similar reasoning is applied for μ_γ and graph \mathbf{Gr}_2 .

For the first player and his \mathbf{Gr}_1 the following cases are possible also:

1. If $\alpha(\lambda) > 0, \beta(\lambda) < 0, \alpha(\lambda) > -\beta(\lambda)$, then
 $\mathbf{Gr}_1 = [(0, 0), (0, y(\lambda))] \cup [(0, y(\lambda)), (1, y(\lambda))] \cup [(1, y(\lambda)), (1, 1)];$
2. If $\alpha(\lambda) < 0, \beta(\lambda) > 0, -\alpha(\lambda) > \beta(\lambda)$, then
 $\mathbf{Gr}_1 = [(0, 1), (0, y(\lambda))] \cup [(0, y(\lambda)), (1, y(\lambda))] \cup [(1, y(\lambda)), (1, 0)];$
3. If $\alpha(\lambda) > 0, \beta(\lambda) < 0, \alpha(\lambda) = -\beta(\lambda)$, then
 $\mathbf{Gr}_1 = [(0, 0), (0, 1)] \cup [(0, 1), (1, 1)];$
4. If $\alpha(\lambda) < 0, \beta(\lambda) > 0, -\alpha(\lambda) = \beta(\lambda)$, then
 $\mathbf{Gr}_1 = [(0, 1), (1, 1)] \cup [(1, 1), (1, 0)];$
5. If $\alpha(\lambda) > 0, \beta(\lambda) = 0$, then $\mathbf{Gr}_1 = [(0, 0), (1, 0)] \cup [(1, 0), (1, 1)];$
6. If $\alpha(\lambda) < 0, \beta(\lambda) = 0$, then $\mathbf{Gr}_1 = [(0, 1), (0, 0)] \cup [(0, 0), (1, 0)];$
7. If $\alpha(\lambda) > 0, \beta(\lambda) < 0, \alpha(\lambda) < -\beta(\lambda)$ or $\alpha(\lambda) < 0, \beta(\lambda) < 0$ or $\alpha(\lambda) = 0, \beta(\lambda) < 0$, then $\mathbf{Gr}_1 = [(0, 0), (0, 1)];$
8. If $\alpha(\lambda) < 0, \beta(\lambda) > 0, -\alpha(\lambda) < \beta(\lambda)$ or $\alpha(\lambda) > 0, \beta(\lambda) > 0$ or $\alpha(\lambda) = 0, \beta(\lambda) > 0$, then $\mathbf{Gr}_1 = [(1, 0), (1, 1)];$
9. If $\alpha(\lambda) = 0, \beta(\lambda) = 0$, then $\mathbf{Gr}_1 = [0, 1] \times [0, 1].$

For the second player the following cases are possible:

1. If $\gamma(\mu) > 0, \delta(\mu) < 0, \gamma(\mu) > -\delta(\mu)$, then
 $\mathbf{Gr}_2 = [(0, 0), (x(\mu), 0)] \cup [(x(\mu), 0), (x(\mu), 1)] \cup [(x(\mu), 1), (1, 1)];$
2. If $\gamma(\mu) < 0, \delta(\mu) > 0, -\gamma(\mu) > \delta(\mu)$, then
 $\mathbf{Gr}_2 = [(0, 1), (x(\mu), 1)] \cup [(x(\mu), 1), (x(\mu), 0)] \cup [(x(\mu), 0), (1, 0)];$
3. If $\gamma(\mu) > 0, \delta(\mu) < 0, \gamma(\mu) = -\delta(\mu)$, then
 $\mathbf{Gr}_2 = [(0, 0), (1, 0)] \cup [(1, 0), (1, 1)];$
4. If $\gamma(\mu) < 0, \delta(\mu) > 0, -\gamma(\mu) = \delta(\mu)$, then
 $\mathbf{Gr}_2 = [(0, 1), (1, 1)] \cup [(1, 1), (1, 0)];$
5. If $\gamma(\mu) > 0, \delta(\mu) = 0$, then $\mathbf{Gr}_2 = [(0, 0), (0, 1)] \cup [(0, 1), (1, 1)];$

6. If $\gamma(\mu) < 0$, $\delta(\mu) = 0$, then $\mathbf{Gr}_2 = [(0, 1), (0, 0)] \cup [(0, 0), (1, 0)]$;
7. If $\gamma(\mu) > 0$, $\delta(\mu) < 0$, $\gamma(\mu) < -\delta(\mu)$ or $\gamma(\mu) < 0$, $\delta(\mu) < 0$ or $\gamma(\mu) = 0$, $\delta(\mu) < 0$, then $\mathbf{Gr}_2 = [(0, 0), (1, 0)]$;
8. If $\gamma(\mu) < 0$, $\delta(\mu) > 0$, $-\gamma(\mu) < \delta(\mu)$ or $\gamma(\mu) > 0$, $\delta(\mu) > 0$ or $\gamma(\mu) = 0$, $\delta(\mu) > 0$, then $\mathbf{Gr}_2 = [(0, 1), (1, 1)]$;
9. If $\gamma(\mu) = 0$, $\delta(\mu) = 0$, then $\mathbf{Gr}_2 = [0, 1] \times [0, 1]$.

Note. For drawing the graphs, the expressions $\alpha(0)$, $\beta(0)$, $y(0)$ and $\alpha(1)$, $\beta(1)$, $y(1)$ are calculated for the first player and $\gamma(0)$, $\delta(0)$, $x(0)$ and $\gamma(1)$, $\delta(1)$, $x(1)$ for the second player. When the player's graph depends only on one of the matrix, it is constructed exactly as in the case of Nash equilibrium [2]. If the expressions $y(0)$ and $y(1)$ do not depend on parameter λ and $y(0) = y(1)$, the graph of the first player will be all the square $[0, 1] \times [0, 1]$. The similar argument is true for the second player.

Based on the above, \mathbf{Gr}_1 and \mathbf{Gr}_2 can be drawn.

The set of Pareto-Nash equilibria (**PNE**) is obtained as the intersection of the player's graphs, that is $\mathbf{PNE} = \mathbf{Gr}_1 \cap \mathbf{Gr}_2$.

Example 2. Consider the following matrices:

$$(A, B) = \begin{pmatrix} 4, 3 & 7, 7 \\ 6, 6 & 8, 4 \end{pmatrix}; (C, D) = \begin{pmatrix} 5, -1 & 2, 4 \\ 4, 3 & 6, 2 \end{pmatrix}.$$

After simplifications, the synthesis functions of the players are:

$$F_1(x, y) = [(5\lambda - 6)y - 4\lambda + 3]x + (4\lambda + 10)y + 4\lambda + 4,$$

$$F_2(x, y) = [(11\mu - 6)x - 3\mu + 1]y + (2\mu + 6)x + 4\mu + 2.$$

In conformity with the described method, the following 4 steps are provided:

1. $\alpha(\lambda) = 5\lambda - 6$, $\beta(\lambda) = -4\lambda + 3$, $y(\lambda) = \frac{4\lambda-3}{5\lambda-6}$; $\gamma(\mu) = 11\mu - 6$, $\delta(\mu) = -3\mu + 1$, $x(\mu) = \frac{3\mu-1}{11\mu-6}$.

2. The values λ_α and μ_γ are the solutions of equations $\alpha(\lambda) = 0$ and $\gamma(\mu) = 0$, respectively. $\lambda_\alpha = \frac{6}{5} \notin (0, 1)$ and $\mu_\gamma = \frac{6}{11} \in (0, 1)$.

3. The values on the interval extremities are calculated:

(a) $y(0) = \frac{1}{2}$, $\alpha(0) = -6 < 0$, $\beta(0) = 3 > 0$ and $-\alpha(0) > \beta(0)$ - the case 2; $y(1) = -1$, $\alpha(1) = -1 < 0$ and $\beta(1) = -1 < 0$ - the case 7. The lines are drawn and the interval between them is hatched. The following result is obtained

$$\mathbf{Gr}_1 = \text{Rectangle} : [(0, 0), (0, y(0)), (1, y(0)), (1, 0)] \cup [(0, 0), (0, 1)],$$

where $y(0) = \frac{1}{2}$.

(b) $x(0) = \frac{1}{6} \in (0, 1)$, $\gamma(0) = -6 < 0$, $\delta(0) = 1 > 0$ and $-\gamma(0) > \delta(0)$ - the case 2; $x(1) = \frac{2}{5} \in (0, 1)$, $\gamma(1) = 5 > 0$, $\delta(1) = -2 < 0$ and $\gamma(1) > -\delta(1)$ - the case 1. The respective lines are drawn and the interval between the respective sides of the square is hatched $[0, 1] \times [0, 1]$. The following result is obtained

$$\mathbf{Gr}_2 = \text{Rectangle} : [(0, 0), (0, 1), (x(0), 1), (x(0), 0)] \cup \text{Rectangle} : [(1, 0), (1, 1), (x(0), 1), (x(0), 0)] \cup [(0, 0), (1, 0)],$$

where $x(0) = \frac{1}{6}$ and $x(1) = \frac{2}{5}$.

4. By determining the intersection of the graphs obtained above, the following set of Pareto-Nash equilibrium in mixed strategies is obtained:

$$\begin{aligned} \mathbf{PNE} &= [(0, 1), (0, 0)] \cup [(0, 0), (1, 0)] \cup \\ &\text{Rectangle} : \left[(0, 0), \left(0, \frac{1}{2}\right), \left(\frac{1}{6}, \frac{1}{2}\right), \left(\frac{1}{6}, 0\right) \right] \cup \\ &\text{Rectangle} : \left[\left(\frac{2}{5}, 0\right), \left(\frac{2}{5}, \frac{1}{2}\right), \left(1, \frac{1}{2}\right), (1, 0) \right]. \end{aligned}$$

6 Wolfram Mathematica Program for Two Criteria Dyadic Games with Mixed Strategies

The method of graph intersection was realized as Wolfram Mathematica Program for Two Criteria Dyadic Games with Mixed Strategies. The program was published on Wolfram Demonstration Project [6]. It may be used online, after the installation of CDF player. The program code may be downloaded at the same address [6], also. The results obtained in the example 2 may be tested online at the same address [6].

7 Concluding remarks

By applying the generalization of the well known notions and by applying the combination of the synthesis function method and the method of intersection of best response graph, the conditions for the Pareto-Nash solutions existence are deduced. The method for determining Pareto-Nash Equilibrium Set in dyadic two criteria games with mixed strategy is clarified from elaboration to final Wolfram Mathematica Program publication. Illustration examples are presented for easier reading. Since the investigated problems have an exponential complexity, a further development of the method in games with bigger dimensions, with implication of the computer science technologies, will be welcome.

References

- [1] V.V. Podinovskii, V.D. Noghin, *Pareto-optimal decisions in multicriteria optimization problems.*, Nauka, Moscow (1982) (in Russian).
- [2] M. Sagaidac, V. Ungureanu, *Operational research.* Chisinau, CEP USM (2004), pp. 178–256.

- [3] V. Ungureanu, *Nash equilibrium set computing in finite extended games*. Computer Science Journal of Moldova, 14 (2006), pp. 345–365.
- [4] V. Ungureanu, *Solution principles for simultaneous and sequential games mixture*. ROMAI Journal, 4 (2008), pp. 225–242.
- [5] V. Lozan, V. Ungureanu, *Principles of Pareto-Nash equilibrium*. Studia Universitatis, 7 (2009), pp. 52–56.
- [6] V. Lozan, V. Ungureanu, *Pareto-Nash Equilibria in Bicriterial Dyadic Games with Mixed Strategies*. <http://demonstrations.wolfram.com/ParetoNashEquilibriaInBicriterialDyadicGamesWithMixedStrateg/>, Wolfram Demonstrations Project, Published: October 13, 2011.

Victoria Lozan, Valeriu Ungureanu,

Received October 23, 2011

Victoria Lozan

State University of Moldova

A. Mateevici str., 60, Chişinău, MD-2009, Republic of Moldova

E-mail: victoria@gmail.com

Valeriu Ungureanu

State University of Moldova

A. Mateevici str., 60, Chişinău, MD-2009, Republic of Moldova

E-mail: v.ungureanu@ymail.com

Analytically determining of the relative inaccuracy (error) of indirectly measurable variable and dimensionless scale characterising quality of the experiment *

Kiril Kolikov, Georgi Krastev†,
Yordan Epitropov, Andrei Corlat

Abstract

In the following paper we present an easily applicable new method for analytical representation of the maximum relative inaccuracy (error) of an indirectly measurable variable $f = f(x_1, x_2, \dots, x_n)$ as a function of the maximum relative inaccuracies (errors) of the directly measurable variables x_1, x_2, \dots, x_n . Our new approach is more adequate for the objective reality. The gist of it is that in order to find the analytical form of the maximum relative inaccuracy of the variable f we take for being fixed variables statistical mean values $\left| \frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1} \right|, \left| \frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2} \right|, \dots, \left| \frac{x_n}{f} \cdot \frac{\partial f}{\partial x_n} \right|$ of the modules of the coefficients of influence of relative inaccuracies $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}$ in f . The numerical value of the maximum relative inaccuracy of the variable f is found using the statistical mean values of the absolute values of the relative inaccuracies $\left| \frac{\Delta x_1}{x_1} \right|, \left| \frac{\Delta x_2}{x_2} \right|, \dots, \left| \frac{\Delta x_n}{x_n} \right|$. Moreover, we look into functions which are continuous but are not differentiable in respect to certain arguments in some points. Having this in mind we develop the theory of errors, which we will call with what we feel is a more precise term – theory of inaccuracies. We introduce some new terms – space of the relative

©2012 by K. Kolikov, G. Krastev†, Y. Epitropov, A. Corlat

*This work was supported by the Fund "Scientific Studies" of the Bulgarian Ministry of Education, Youth and Science as part of the contract DTK 02/35

inaccuracy and plane of the relative inaccuracy of f . We also define a sample plane of the ideal absolutely accurate experiment and using it we define a universal numerical characteristic – a dimensionless scale for evaluation of the quality (accuracy) of the experiment.

Keywords: Indirectly measurable variable, maximum relative error, dimensionless scale.

1 Introduction

Many natural and social processes are described by indirectly measurable variables dependent on a finite number of directly measurable variables. As it is known, the measuring of the values of each directly measurable variable is accompanied by inaccuracies. With respect to their alteration characteristics, the inaccuracies are systematic or random [1].

The systematic inaccuracies are permanent or their alteration can be described by a law. They are the result of certain constant influences which cannot be foreseen. Towards the random accuracies we shall also account what we call *hidden inaccuracies* that are the result of influences of measured active objects [2] that appear in a time interval and change the natural progress of the observed process (such as objects that have their own sources of energy; living organisms with their own will, etc.).

Thus each measurable variable is determined with a total inaccuracy that is caused by both systematic and random inaccuracies. Therefore, it is of great importance to develop a reliable method for finding the total inaccuracy of given measurable variable.

In respect of the way a certain inaccuracy is represented it can be absolute or relative. An absolute inaccuracy is expressed in units of the measured variable. A relative inaccuracy is a dimensionless variable and is represented by the ratio of the absolute inaccuracy and the value of the measured experimental variable.

Usually the processes that are being studied are mathematically modelled with real functions that are differentiable in their domains.

(Even non-continuous functions can be viewed as differentiable in a certain sense.)

Here is a real differentiable function

$$f = f(x_1, x_2, \dots, x_n) \neq 0 \quad (1)$$

of n real independent variables x_1, x_2, \dots, x_n , which help one to be able to model directly measurable variables (with the help of measuring tools or methods). Then the function f models one indirectly measurable variable.

In order to calculate the absolute and the relative inaccuracies of indirectly measurable variable f which has continuous first partial derivatives in respect to all its variables there are two principles [3, 4, 5] given in scientific and academic literature.

The first principle gives the maximum absolute inaccuracy Δf of the function f . Initially we have to determine the full differential

$$df = \sum_{i=1}^n \frac{\partial f}{\partial x_i} dx_i \quad (2)$$

of the function (1). When the inaccuracies of the measurements are small enough in formula (2) the differential d can be replaced with the finite difference Δ , and in this substitution every minus is replaced with a plus to reach the maximum value of the inaccuracy. Thus one obtains

$$\Delta f = \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \cdot |\Delta x_i|, \quad (3)$$

where Δx_i is the maximum absolute inaccuracy of the directly measurable variable $x_i (i = 1, 2, \dots, n)$. The maximum relative inaccuracy f_r is then determined by the expression

$$f_r = \frac{\Delta f}{|f|} = \frac{1}{|f|} \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \cdot |\Delta x_i|. \quad (4)$$

By the second principle, we initially find the maximum relative inaccuracy f_r of the function f . In order to do this, we find the logarithm of the function (1) and then we determine full differential of the result. Further we replace the differential d in the same way with the finite difference Δ and again replace every minus with a plus to reach the maximum value of the inaccuracy. Thus, we get the maximum relative inaccuracy f_r . Then the maximum absolute inaccuracy Δf is determined from the expression $\Delta f = f \cdot f_r$.

Since in the set of real numbers we can find only the logarithms of positive variables, the second principle limits the class of functions for which we can find the respective inaccuracies.

In [1], based solely on differentiation we define a simple method for representing the maximum absolute (total) inaccuracy of an indirectly measurable variable $f(x_1, x_2, \dots, x_n)$ as a function of the maximum absolute (total) inaccuracies of the directly measurable variables x_1, x_2, \dots, x_n . Based on this method we introduce a numerical characteristic – dimensionless scale for evaluation of the quality (accuracy) of the experiment.

The purpose of this paper is to apply this procedure for the maximum relative inaccuracy of $f(x_1, x_2, \dots, x_n)$. Moreover, we extend the type of the function f by also looking into the case when it is not differentiable but is continuous in respect to some arguments in some points.

2 Analytic representation of the maximum relative inaccuracy of an indirectly measurable variable

Firstly, let the function f has continuous partial derivatives in respect to all its variables. Let us present formula (4) in this way

$$f_r = \sum_{i=1}^n \left| \frac{x_i}{f} \cdot \frac{\partial f}{\partial x_i} \right| \cdot \left| \frac{\Delta x_i}{x_i} \right| \quad (x_i \neq 0, i = 1, 2, \dots, n). \quad (5)$$

This shows that evaluation of the relative inaccuracy of the indi-

rectly measurable variable $f(x_1, x_2, \dots, x_n)$ is dependant not only on the relative inaccuracies with which the directly measureable variables x_1, x_2, \dots, x_n are determined, but also on the analytical form of the function f itself. For $i = 1, 2, \dots, n$ every addend of the form $\left| \frac{x_i}{f} \cdot \frac{\partial f}{\partial x_i} \right| \cdot \left| \frac{\Delta x_i}{x_i} \right|$ is the partial relative inaccuracy of the result of the indirect measurement of the function f , caused by the inaccuracy $\frac{\Delta x_i}{x_i}$ with which the variable x_i is determined.

The variable $\left| \frac{x_i}{f} \cdot \frac{\partial f}{\partial x_i} \right|$ is in fact the coefficient of influence of the inaccuracy $\frac{\Delta x_i}{x_i}$ when determining the relative inaccuracy of $f(x_1, x_2, \dots, x_n)$.

Let us now assume the function $f(x_1, x_2, \dots, x_n)$ is not differentiable in respect to its argument x_k ($1 \leq k \leq n$) in some points a_{kj} , but is continuous in respect to this argument in a_{kj} . The partial derivative $\frac{\partial f}{\partial x_k}$ does not exist in the points a_{kj} .

However, if for f there are right and left derivatives given $x_k \rightarrow a_{kj}$, then in order to compute the maximum relative inaccuracy in the formula (5) the variable $\frac{\partial f}{\partial x_k}$ is replaced by the one of the two limits in which the coefficient of influence $\left| \frac{x_k}{f} \cdot \frac{\partial f}{\partial x_k} \right|$ has greater value. (The values of the two limits are different, otherwise f would be differentiable in respect to x_k). An example for such function would be $f = A \arcsin(\sin x)$ representing the voltage of triangle signal [6]. In Figure 1 we have shown the graphics of the function when $A = 500$ V and $x = 2\pi vt \in \left[0, \frac{13}{4}\pi\right]$, where $v = 15$ kHz is the linear frequency and t is the time.

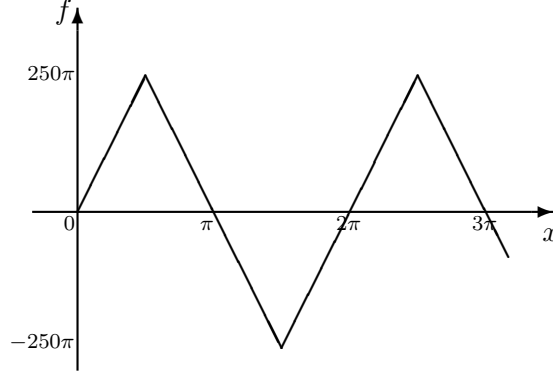


Figure 1. Graphics of the function $f = 500 \arcsin(\sin x)$, $x \in \left[0, \frac{13}{4}\pi\right]$

We would like to point out that this procedure for indifferentiable but continuous functions in respect to its arguments in some points is proposed for the first time in the theory of inaccuracies.

Moreover, a novelty in our approach for determining the analytical form of the maximum relative (total) inaccuracy of the indirectly measurable variable $f(x_1, x_2, \dots, x_n)$ is that in formula (5) we take for fixed variables the mean values $\left|\frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1}\right|, \left|\frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2}\right|, \dots, \left|\frac{x_n}{f} \cdot \frac{\partial f}{\partial x_n}\right|$ of the absolute values of the coefficients of influence of the relative inaccuracies $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}$ of the indirectly measurable variable f and the maximum relative inaccuracies $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}$ of the directly measurable variables x_1, x_2, \dots, x_n are considered to be variables.

Thus according to (5) *the maximum relative inaccuracy f_r of the indirectly measurable variable f is a linear function of the maximum relative inaccuracies $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}$ of the directly measurable variables x_1, x_2, \dots, x_n .*

If we look at $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}, \pm f_r$ as a system of generalized orthogonal coordinates, for $n > 2$ we get an $n + 1$ -dimensional metric hyperspace F_r^{n+1} , where (5) is the equation of a hyperplane that passes through the origin of the coordinate system. The hyperspace F_r^{n+1} will

be called *space of the relative inaccuracy of f* , and f_r will be called *plane of the relative inaccuracy of f* .

For $n = 2$, according to formula (5) we have

$$f_r = \left| \frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1} \right| \cdot \left| \frac{\Delta x_1}{x_1} \right| + \left| \frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2} \right| \cdot \left| \frac{\Delta x_2}{x_2} \right|.$$

Thus the equation $f_r = f_r \left(\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2} \right)$ is an equation of the plane of the relative inaccuracy in the three-dimensional metric space F_r^3 of the relative inaccuracy of f .

For $n = 1$ formula (5) becomes

$$f_r = \left| \frac{x}{f} \cdot \frac{\partial f}{\partial x} \right| \cdot \left| \frac{\Delta x}{x} \right|.$$

Thus the equation $f_r = f_r \left(\frac{\Delta x}{x} \right)$ is an equation of the line of the relative inaccuracy in the two-dimensional metric space F_r^2 of the relative inaccuracy of f .

The term space of the relative inaccuracy is introduced for the first time in this paper. Moreover, as we pointed in [1], in a certain sense it is an analogy of the imaginary configurative space in the Hamiltonian reformulation of the classical mechanics [7]. In that same sense the system $\frac{\Delta x_1}{x_1}, \frac{\Delta x_2}{x_2}, \dots, \frac{\Delta x_n}{x_n}, \pm f_r$ can be viewed as generalised orthogonal coordinates.

3 Determining the numerical value of the maximum relative inaccuracy of an indirectly measurable variable

Let us have an experiment where k measurements of the directly measurable variables x_1, x_2, \dots, x_n are made. On the m -th measurement ($m = 1, 2, \dots, k$) the absolute values of the coefficients of influence

$\left| \frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1} \right|_m, \left| \frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2} \right|_m, \dots, \left| \frac{x_n}{f} \cdot \frac{\partial f}{\partial x_n} \right|_m$ and of the relative inaccuracies $\left| \frac{\Delta x_1}{x_1} \right|_m, \left| \frac{\Delta x_2}{x_2} \right|_m, \dots, \left| \frac{\Delta x_n}{x_n} \right|_m$ are calculated. After this the mean values $\overline{\left| \frac{x_j}{f} \cdot \frac{\partial f}{\partial x_j} \right|} = \frac{1}{k} \sum_{m=1}^k \left| \frac{x_j}{f} \cdot \frac{\partial f}{\partial x_j} \right|_m$ ($j = 1, 2, \dots, n$) are calculated and from formula (5) one can get the analytical representation (equation) of the plane of the inaccuracies.

Furthermore, if $\overline{\left| \frac{\Delta x_i}{x_i} \right|} = \frac{1}{k} \sum_{m=1}^k \left| \frac{\Delta x_i}{x_i} \right|_m$, then according to formula (5) the numerical value of the maximum relative inaccuracy

$$f_r = \sum_{i=1}^n \overline{\left| \frac{x_i}{f} \cdot \frac{\partial f}{\partial x_i} \right|} \cdot \overline{\left| \frac{\Delta x_i}{x_i} \right|}$$

is determined, as the point $\left(\overline{\left| \frac{\Delta x_1}{x_1} \right|}, \overline{\left| \frac{\Delta x_2}{x_2} \right|}, \dots, \overline{\left| \frac{\Delta x_n}{x_n} \right|}, f_r \right)$ lies in the plane of the relative inaccuracy.

The function $f(x_1, x_2, \dots, x_n)$ can be considered as a random variable of random independent variables. In that sense the suggested by us method for computing f_r is more adequate to the objective reality because the statistical mean value of a random variable is actually its most probable value. Again in that sense the plane of the relative inaccuracy of f is a *stochastic plane*.

The numerical value of the maximum absolute inaccuracy Δf of the experiment can be determined directly using [1] or using formula (4) and the known numerical value of f_r .

If the function $f(x_1, x_2, \dots, x_n)$ is not differentiable in respect to an argument x_k ($1 \leq k \leq n$) in some points a_{kj} , but is continuous in respect to this argument in a_{kj} , then the method is applied to the maximum absolute inaccuracy analogically to the already described way.

4 Scale characterising the quality of the experiment

It is very important and advantageous for every measuring method to have a numerical characteristic – a scale which is used to evaluate the quality of the experiment, i. e. its accuracy. For the first time in the theory of the inaccuracies we suggested this kind of scale in [1]. Here we suggest analogical scale which has the important property *dimensionless*, i. e. the quality of the experiment is expressed only with a number, not with the units of measurements.

Let us look at the stochastic plane α of the relative inaccuracy of f . According to (5) its general equation is of the following type

$$\alpha : \sum_{i=1}^n A_i \cdot \left| \frac{\Delta x_i}{x_i} \right| - f_r = 0,$$

where $A_i = \left| \frac{x_i}{f} \cdot \frac{\partial f}{\partial x_i} \right| = \text{const} \geq 0$. As we have already emphasised, this is the equation of a hyperplane in the hyperspace F_r^{n+1} going through the beginning of the coordinate system.

Let us also take a look at the hyperplane

$$\varepsilon : f_r = 0.$$

It is obvious that the equation $f_r = 0$ is possible if and only if $\frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1} = \frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2} = \dots = \frac{x_n}{f} \cdot \frac{\partial f}{\partial x_n} = 0$, i. e. if and only if $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \dots = \frac{\partial f}{\partial x_n} = 0$. Thus we take ε for *sample plan in the space of the relative inaccuracy* which represents an imaginary ideal perfectly accurate experiment even though this experiment is impossible and the sample plane ε is unreachable. However, by increasing the accuracy of the real experiment the plane α approximates ε . Thus the smaller the deviation of the plane α of the experiment from the sample plane ε of the ideal experiment is, i. e. the smaller the angle between these two planes is, the more accurate the experiment is. This angle

can be always calculated as it is equal to the angle between the normal vectors $\vec{n}_\alpha(A_1, A_2, \dots, A_n, -1)$ of the plane α and $\vec{n}_\varepsilon(0, 0, \dots, 0, -1)$ of the plane ε . Then the value of the cosine

$$k_\alpha = \cos \angle(\vec{n}_\alpha, \vec{n}_\varepsilon) = \frac{1}{\sqrt{A_1^2 + A_2^2 + \dots + A_n^2 + 1}} \quad (6)$$

of this angle can be chosen for a *coefficient of accuracy* in a *dimensionless scale*, i. e. for a *numerical characteristic of the quality of the experiment*.

The scale for evaluating the quality of the experiment is the interval $[0, 1]$. An experiment is as accurate as the value of the coefficient of accuracy k_α is closer to 1 and is as inaccurate as the value of the coefficient of accuracy k_α is closer to 0. The value $k_\alpha = 1$ represents the ideal perfectly accurate experiment and the value $k_\alpha = 0$ – the ideal absolutely inaccurate experiment.

Then from formula (6) we get the following

Criterion for accuracy of an experiment – *An experiment is as accurate as possible if and only if the sum of the squares of the coefficients*

$$A_1^2 + A_2^2 + \dots + A_n^2 = \left(\frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1} \right)^2 + \left(\frac{x_2}{f} \cdot \frac{\partial f}{\partial x_2} \right)^2 + \dots + \left(\frac{x_n}{f} \cdot \frac{\partial f}{\partial x_n} \right)^2$$

is the least possible.

The accuracy of the experiment can, of course, be interpreted using the angle between the stochastic plane α of the relative inaccuracy and the sample plane ε . Then the scale for evaluating the quality of the experiment is the interval $\left[0, \frac{\pi}{2}\right]$. An experiment is as accurate as the value of $\arccos k_\alpha$ is closer to 0, and is as inaccurate as the value of $\arccos k_\alpha$ is closer to $\frac{\pi}{2}$. The value 0 corresponds to the ideal perfectly accurate experiment and the value $\frac{\pi}{2}$ – to the ideal absolutely inaccurate experiment.

We take for basic scale the interval $[0, 1]$ and for basic measurement of the accuracy of the experiment the value of the coefficient of accuracy $k_\alpha = \cos \angle(\vec{n}_\alpha, \vec{n}_\varepsilon)$ since both of them are dimensionless variables contrary to the second scale and measurement which are measured in radians.

5 An example and computations

We will present an example which illustrates our method and scale. It is known that the coefficient η of the viscosity of a liquid with density ρ can be determined using the Stokes' method when a sphere with radius r and density ρ_1 is put in a cylindrical container filled in with the examined liquid. In a given moment of time the sphere starts to descent steadily with a constant speed ν and its weight is the same as the buoyancy and the force of internal friction (viscosity) of the liquid as the following holds $\eta = \frac{2r^2g}{9\nu}(\rho_1 - \rho)$, where $g = 9.8 \text{ m}\cdot\text{s}^{-2}$ is the gravity of Earth.

We made an experiment for measuring the viscosity of glycerine (under temperature $t = 18^\circ \text{ C}$) using lead spheres having measured the density of the glycerine and the lead in advance. In Table 1 experimental data from the measurements and the corresponding calculated values of the viscosity with accuracy of four digits in the decimal part are given.

Table 1. Experimental data from the measurements of the viscosity of glycerine using the Stokes' method.

m -th mea- sure- ment	$r_m[\text{m}]$	$\nu_m[\text{m}\cdot\text{s}^{-1}]$	$\rho_m[\text{kg}\cdot\text{m}^{-3}]$	$\rho_{1m}[\text{kg}\cdot\text{m}^{-3}]$	$\eta_m[\text{Pa}\cdot\text{s}]$
1	5×10^{-4}	4.54×10^{-3}	1.262×10^3	1.1341×10^4	1.2087
2	5×10^{-4}	4.52×10^{-3}	1.262×10^3	1.1341×10^4	1.214
3	5×10^{-4}	4.5×10^{-3}	1.262×10^3	1.1341×10^4	1.2194
4	4.8×10^{-4}	4.22×10^{-3}	1.26×10^3	1.134×10^4	1.1944
5	4.8×10^{-4}	4.2×10^{-3}	1.26×10^3	1.134×10^4	1.2003

5.1 Computing the maximum relative inaccuracy η_r , using the classical method

According to (4) we have

$$\eta_r = \left| \frac{2}{r} \right| \cdot |\Delta r| + \left| \frac{1}{\nu} \right| \cdot |\Delta \nu| + \left| \frac{1}{\rho_1 - \rho} \right| \cdot |\Delta \rho| + \left| \frac{1}{\rho_1 - \rho} \right| \cdot |\Delta \rho_1|.$$

We firstly compute the mean values

$$\bar{r} = \frac{1}{5} \sum_{m=1}^5 r_m = 4.92 \times 10^{-4} \text{ m},$$

$$\bar{\nu} = \frac{1}{5} \sum_{m=1}^5 \nu_m = 4.396 \times 10^{-3} \text{ m} \cdot \text{s}^{-1},$$

$$\bar{\rho} = \frac{1}{5} \sum_{m=1}^5 \rho_m = 1.2612 \times 10^3 \text{ kg} \cdot \text{m}^{-3},$$

$$\bar{\rho}_1 = \frac{1}{5} \sum_{m=1}^5 \rho_{1m} = 1.13406 \times 10^4 \text{ kg} \cdot \text{m}^{-3}.$$

Then $|\Delta r|_1 = |r_1 - \bar{r}| = 0.08 \times 10^{-4} \text{ m}$, $|\Delta \nu|_1 = |\nu_1 - \bar{\nu}| = 0.144 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$, $|\Delta \rho|_1 = |\rho_1 - \bar{\rho}| = 0.0008 \times 10^3 \text{ kg} \cdot \text{m}^{-3}$, $|\Delta \rho_1|_1 = |\rho_{11} - \bar{\rho}_1| = 0.00004 \times 10^4 \text{ kg} \cdot \text{m}^{-3}$ and thus we get the value of

$$\eta_{r,1} = \frac{2}{r_1} \cdot |\Delta r|_1 + \frac{1}{\nu_1} \cdot |\Delta \nu|_1 + \frac{1}{\rho_{11} - \rho_1} \cdot |\Delta \rho|_1 + \frac{1}{\rho_{11} - \rho_1} \cdot |\Delta \rho_1|_1 = 0.0779.$$

Analogically, we compute $\eta_{r,2} = 0.0737$, $\eta_{r,3} = 0.0694$, $\eta_{r,4} = 0.1172$,

$\eta_{r,5} = 0.1224$. We finally get $\eta_r = \frac{1}{5} \sum_{m=1}^5 \eta_{r,m} = 0.0921$.

5.2 Computing the maximum relative inaccuracy η_r , using the method introduced by us

According to Section 2 under the same measurements we compute the absolute values of the coefficients of influence $\left| \frac{r}{\eta} \cdot \frac{\partial \eta}{\partial r} \right| = 2$, $\left| \frac{\nu}{\eta} \cdot \frac{\partial \eta}{\partial \nu} \right| = 1$, $\left| \frac{\rho}{\eta} \cdot \frac{\partial \eta}{\partial \rho} \right|_m = \frac{\rho_m}{\rho_{1m} - \rho_m}$, $\left| \frac{\rho_1}{\eta} \cdot \frac{\partial \eta}{\partial \rho_1} \right|_m = \frac{\rho_{1m}}{\rho_{1m} - \rho_m}$ and the ones of the relative inaccuracies $\left| \frac{\Delta r}{r} \right|_m$, $\left| \frac{\Delta \nu}{\nu} \right|_m$, $\left| \frac{\Delta \rho}{\rho} \right|_m$, $\left| \frac{\Delta \rho_1}{\rho_1} \right|_m$. The results can be seen in Table 2.

Table 2. The values of the non-constant coefficients of influence and the relative inaccuracies of the variables in the experiment

m -th mea- sure- ment	$\left \frac{\rho}{\eta} \cdot \frac{\partial \eta}{\partial \rho} \right _m$	$\left \frac{\rho_1}{\eta} \cdot \frac{\partial \eta}{\partial \rho_1} \right _m$	$\left \frac{\Delta r}{r} \right _m$	$\left \frac{\Delta \nu}{\nu} \right _m$	$\left \frac{\Delta \rho}{\rho} \right _m$	$\left \frac{\Delta \rho_1}{\rho_1} \right _m$
1	0.1252	1.1252	0.016	0.0317	6.339×10^{-4}	3.52×10^{-5}
2	0.1252	1.1252	0.016	0.0274	6.339×10^{-4}	3.52×10^{-5}
3	0.1252	1.1252	0.016	0.0231	6.339×10^{-4}	3.52×10^{-5}
4	0.125	1.125	0.025	0.0417	9.523×10^{-4}	5.29×10^{-5}
5	0.125	1.125	0.025	0.0467	9.523×10^{-4}	5.29×10^{-5}

We compute the statistical mean values

$$\overline{\left| \frac{\rho}{\eta} \cdot \frac{\partial \eta}{\partial \rho} \right|} = \frac{1}{5} \sum_{m=1}^5 \left| \frac{\rho}{\eta} \cdot \frac{\partial \eta}{\partial \rho} \right|_m = 0.1251,$$

$$\overline{\left| \frac{\rho_1}{\eta} \cdot \frac{\partial \eta}{\partial \rho_1} \right|} = \frac{1}{5} \sum_{m=1}^5 \left| \frac{\rho_1}{\eta} \cdot \frac{\partial \eta}{\partial \rho_1} \right|_m = 1.1251.$$

Then according to (5) the analytical form of η_r is

$$\eta_r = 2 \left| \frac{\Delta r}{r} \right| + \left| \frac{\Delta \nu}{\nu} \right| + 0.1251 \left| \frac{\Delta \rho}{\rho} \right| + 1.1251 \left| \frac{\Delta \rho_1}{\rho_1} \right|.$$

Following Section 3 we compute the statistical mean values $\overline{\left|\frac{\Delta r}{r}\right|} = \frac{1}{5} \sum_{m=1}^5 \left|\frac{\Delta r}{r}\right|_m = 0.0196$, $\overline{\left|\frac{\Delta \nu}{\nu}\right|} = \frac{1}{5} \sum_{m=1}^5 \left|\frac{\Delta \nu}{\nu}\right|_m = 0.0341$, $\overline{\left|\frac{\Delta \rho}{\rho}\right|} = \frac{1}{5} \sum_{m=1}^5 \left|\frac{\Delta \rho}{\rho}\right|_m = 7.6126 \times 10^{-4}$ and $\overline{\left|\frac{\Delta \rho_1}{\rho_1}\right|} = \frac{1}{5} \sum_{m=1}^5 \left|\frac{\Delta \rho_1}{\rho_1}\right|_m = 4.228 \times 10^{-5}$. For numerical value of the maximum relative inaccuracy we get $\eta_r = 2 \left|\frac{\Delta r}{r}\right| + \left|\frac{\Delta \nu}{\nu}\right| + 0.1251 \left|\frac{\Delta \rho}{\rho}\right| + 1.1251 \left|\frac{\Delta \rho_1}{\rho_1}\right| = 0.0734$.

We can see that there is a certain difference between the classic method and our method which gives more adequate to the reality results.

Furthermore, let $A_1 = \overline{\left|\frac{\partial \eta}{\partial r}\right|} = 2$, $A_2 = \overline{\left|\frac{\partial \eta}{\partial \nu}\right|} = 1$, $A_3 = \overline{\left|\frac{\partial \eta}{\partial \rho}\right|} = 0.1251$ and $A_4 = \overline{\left|\frac{\partial \eta}{\partial \rho_1}\right|} = 1.1251$. According to Section 4 the stochastic plane α of the relative inaccuracy of η in the space $\left(\frac{\Delta r}{r}, \frac{\Delta \nu}{\nu}, \frac{\Delta \rho}{\rho}, \frac{\Delta \rho_1}{\rho_1}, \pm \eta_r\right)$ of the relative inaccuracy has general equation $\alpha : A_1 \left|\frac{\Delta r}{r}\right| + A_2 \left|\frac{\Delta \nu}{\nu}\right| + A_3 \left|\frac{\Delta \rho}{\rho}\right| + A_4 \left|\frac{\Delta \rho_1}{\rho_1}\right| - \eta_r = 0$, i. e.

$$\alpha : 2 \left|\frac{\Delta r}{r}\right| + \left|\frac{\Delta \nu}{\nu}\right| + 0.1251 \left|\frac{\Delta \rho}{\rho}\right| + 1.1251 \left|\frac{\Delta \rho_1}{\rho_1}\right| - \eta_r = 0.$$

According to (6) the value of the coefficient of accuracy of the experiment is

$$k_\alpha = \cos \angle(\vec{n}_\alpha, \vec{n}_\epsilon) = \frac{1}{\sqrt{A_1^2 + A_2^2 + A_3^2 + A_4^2 + 1}} = 0.3706.$$

Conclusion 1. The experiment is not very accurate because the coefficient of accuracy k_α is closer to 0, not 1. It is important to point out that this does not necessarily mean that the experimental data for the directly measurable variables are very inaccurate. But when these conditions are met, small alterations of the values of the variables $\frac{\Delta \rho}{\rho}$

and $\frac{\Delta\rho_1}{\rho_1}$ lead to substantial alterations of the values of the function $\eta_r = \eta_r \left(\frac{\Delta r}{r}, \frac{\Delta\nu}{\nu}, \frac{\Delta\rho}{\rho}, \frac{\Delta\rho_1}{\rho_1} \right)$.

Conclusion 2. The Criterion for accuracy from Section 4 gives conditions under which the accuracy of the experiment can be increased, namely: to select such values of the directly measureable variables ρ and ρ_1 , under which the values of the coefficients of influence $\frac{\rho}{\eta} \cdot \frac{\partial\eta}{\partial\rho} = \frac{\rho}{\rho_1 - \rho}$ and $\frac{\rho_1}{\eta} \cdot \frac{\partial\eta}{\partial\rho_1} = \frac{\rho_1}{\rho_1 - \rho}$ are smaller, i. e. the accuracy of the experiment can be increased if the sphere which is put in the glycerine is with higher density. It is important to point out that if that condition was met, this does not mean that the experimental data for ρ and ρ_1 were going to be more accurate, but that small alterations of the values of the variables $\frac{\Delta\rho}{\rho}$ and $\frac{\Delta\rho_1}{\rho_1}$ would lead to small alterations of the values of the function $\eta_r = \eta_r \left(\frac{\Delta r}{r}, \frac{\Delta\nu}{\nu}, \frac{\Delta\rho}{\rho}, \frac{\Delta\rho_1}{\rho_1} \right)$.

6 Discussion

The advantages of the presented in this paper method for analytically representing the maximum relative inaccuracy of an indirectly measurable variable and for computing its value can be summarised in the following basic directions.

- (i) *More adequate* to the objective reality *quantity value* of the maximum relative inaccuracy of the indirectly measurable variable.
- (ii) Using the Criterion of the accuracy of the experiment, the method shows *conditions under which the accuracy* of the experiment *is the greatest possible one*.
- (iii) *Universality*, because this method can be applied in different scientific fields, in experiments held using various utensils and methods, in mathematical models described even with indifferential functions.
- (iv) *Clarity* and *observability* of the results when $n = 1$ or $n = 2$ using a computer generated graphical representation, accordingly, using a line or a plane.

We have to point out that natural processes are described by continuous functions which are either differentiable or indifferentiable in some points, but there are left and right derivatives in these points. However, in mathematical models describing these processes this might not be the case. In these models the method is not applicable.

Each utensil for measurement measures a real and not mathematically modeled variable. The real variables are always finite and no matter how fast (or even explosive) their alteration is, it is never (strictly) jumping but rather smooth in a small enough interval of time. The representations of jumps in the behavior of a function and of an infinite value of a real variable are only theoretical (model). They are reasonable abstractions when describing the objective reality.

Moreover, the dimensionless scale of the quality of an experiment gives an opportunity for:

- (i) *Quality evaluation* of the experiment;
- (ii) *Comparison between the efficiency* of different experimental methods in one research can be compared. Moreover, the method makes it possible to even compare the efficiency of experimental methods from different scientific fields.

7 Conclusion

While in the classical method the mean arithmetic values of the indirectly measurable variable f_r are used, in our method we use the statistical mean values of the random variables that f is composed of. Thus we get the most probable value for f_r .

Moreover, in practice the maximum relative inaccuracy of a measurable variable finds a much wider application than the maximum absolute relative inaccuracy because it is a dimensionless variable and can be presented in percentages.

The suggested by us method is of great importance for every experimental science – physics, chemistry, biology, medicine, sociology, economics, etc. in which the studied processes are described by differentiable functions. Using it, not only more adequate to the reality numerical value of the maximum relative inaccuracy can be determined

given a certain experiment, but also using the dimensionless scale a quantity evaluation of the quality (accuracy) of the experiment can be given and the conditions, under which this accuracy can be increased, can be determined. The dimensionless scales used in the experimental science have certain advantages compared to the unit ones. An important one is that results from measurements of essentially different variables can be compared.

References

- [1] K. Kolikov, G. Krastev, Y. Epitropov, D. Hristozov. *Analytically Determining of the Absolute Inaccuracy (Error) of Indirectly Measurable Variable and Dimensionless Scale Characterising the Quality of the Experiment*. Chemometr Intell Lab, 102 (2010), pp. 15–19.
- [2] K. Kolikov, G. Krustev. *Hidden Parameters in the Laws of Motion of Material Points*. Scientific Research of the Union of Scientists in Bulgaria – Plovdiv, series C. Technics and Technologies, Vol. VII., Union of Scientists, 60th Anniversary Jubilee Scientific Session, November 4-5, 2008, pp. 232–237.
- [3] J. Epperson. *An Introduction to Numerical Methods and Analysis*. Wiley-Interscience, 2007.
- [4] G. Squires. *Practical Physics*. Cambridge University Press, 2008.
- [5] J. Taylor. *An Introduction to Error Analysis: the Study of Uncertainties in Physical Measurements*. University Science Books, 1997.
- [6] P. Horowitz, W. Hill. *The Art of Electronics*. Cambridge University Press, 1989.
- [7] H. Goldstein, C. Poole, J. Safko. *Classical Mechanics*. Addison Wesley, 2001.

Kiril Kolikov, Georgi Krastev†,
Yordan Epitropov, Andrei Corlat

Received November 14, 2011

Kiril Kolikov
Plovdiv University "P. Hilendarski"
24 Tzar Asen Street, 4000 Plovdiv, Bulgaria
Phone: +359 886966562
E-mail: *kolikov@uni-plovdiv.bg*

Yordan Epitropov
Plovdiv University "P. Hilendarski"
24 Tzar Asen Street, 4000 Plovdiv, Bulgaria
Phone: +359 888854943
E-mail: *epitropov@uni-plovdiv.bg*

Andrei Corlat
Moldova State University
60 Alexei Mateevici Street, MD 2009, Chisinau, Moldova
Phone: +373 69173271
E-mail: *an.corlat@gmail.com*

Mathematical modeling of high-speed loads effects on underground storage tanks*

Elena Gutuleac

Abstract

The purpose of this paper is to provide a numerical modeling of intense dynamic loads on engineering materials. Numerical results illustrate the evolution of the state of elastic-plastic shells (filled with fluid or elastic-plastic material), which are subjected to explosive loads.

Keywords: numerical modeling, elastic-plastic media, high-speed loads, numerical method.

1 Introduction

The elements of thin-wall constructions are used in various fields of engineering practice and most of them function in conditions of rather high operational load [1-3]. These objects include different shell containers constructed of materials with diverse physical and mechanical properties and intended to store flammable, toxic and chemical substances. Their stress-strain state exposed to a wide range of external forces and force-majeure circumstances represents a significant interest for analysis in order to level down the accidental risk and environmental impact. In this paper the behavior of shell containers deepened in ground and exposed to intense dynamic loads is considered [4, 5]. To adequately describe all the processes arising under intense dynamic loads of the material it is important to choose a particular mathematical model of the environment. This model should take into account a noninvertible nature of deformation, the dependence on strain rates

and other processes associated with an explosive loading. The behavior of various materials is described within the framework of state equation in the form of Mie-Gruneisen, taking into account the complex stress-strain behavior of the matter. The ground is represented as a three-component substance consisting of solid granules, water and air [3]. Computer modeling of deformation under explosive loading supposes setting a numerical method and obtaining a picture of internal parameters' evolution. In this paper following Wilkins [6-8] we consider a second order accurate finite-difference scheme and provide its modification to solving specific problems. The use of adaptive meshes, parallel numerical algorithms and multiprocessor clusters allows one to reduce computing time [9].

2 Setting of the problem and mathematical model

Assume an underground container is subjected to explosive loading. Note that explosives are situated inside and/or outside the shell. The behavior of these structures under intense dynamic loads is of unsteady nature and is described with the help of environment models and physical laws of loading. Let us examine the dynamics of explosive loading within the framework of two-dimensional model of elastic-plastic medium and solve the following basic equations [6]:

$$\begin{aligned}
 \sigma' &= k_0 \left(\varepsilon_{kk} - \alpha_\nu (T - T_0) - \frac{\Lambda}{3} \int_0^\omega \frac{\partial \dot{\omega}}{\partial \sigma} \partial \omega \right) \\
 (\tau'_{ij})^\nabla + \lambda \tau'_{ij} &= 2\mu_0 \dot{\varepsilon}_{ij}, \tau'_{ij} \tau'_{ij} \leq \frac{2}{3} Y_0^2, \\
 \rho c_0 \dot{T} + \alpha_\nu \dot{\sigma} T &= \tau_{ij} \dot{\varepsilon}_{ij}^p + \Lambda \dot{\omega}^2 - \text{div} \bar{q}, \\
 \dot{\omega} &= B(\sigma' - \sigma_*)^m H(\sigma' - \sigma_*), \\
 \tau_{ij} &= S_{ij} + \Gamma \varepsilon_{ij}, \quad \tau'_{ij} = \tau_{ij} / (1 - \omega), \sigma' = \sigma / (1 - \omega).
 \end{aligned} \tag{1}$$

Here the symbol T denotes temperature; ρ – density; \bar{q} – heat transfer rate; $\sigma_{ij} = \sigma \delta_{ij} + s_{ij}$ – stress component, divided into two mutually

orthogonal tensors, spherical tensor $\sigma\delta_{ij} = \sigma_{kk}\delta_{ij}/3$ and deviator s_{ij} ; $\varepsilon_{ij}, \varepsilon_{ij}^e, \varepsilon_{ij}^p$ – elastic and plastic strain components; $e_{ii} = \varepsilon_{ij} - 1/3\varepsilon_{kk}\delta_{ij}$ – strain deviator components; ω – structural parameter describing the origin and growth of material's vulnerability; $H(x)$ – Heviside function; δ_{ii} – Kronecker delta; k_0 and μ_0 – bulk modulus and shear modulus of undamaged material; α_ν – cubic expansion coefficient; c_σ – heat at constant pressure; τ_{ij} – components of the "active" stress tensor; A, B, m, Γ – material's characteristics. We assume $\varepsilon_{ij} = \varepsilon_{ij}^e + \varepsilon_{ij}^p$ and $\varepsilon_{kk}^p = 0$, then plastic flow is incompressible. ∇ is a Yauman's derivative of tensor components:

$$S_{ij}^\nabla = \dot{S}_{ij} - S_{ik}\omega_{jk} - S_{jk}\omega_{ik}; \quad (2)$$

$$\omega'_{ij} = 1/2\left(\frac{\partial\nu_i}{\partial x_j} - \frac{\partial\nu_j}{\partial x_i}\right),$$

ν – velocity components, x – Cartesian coordinates; λ is determined by the Mises plastic's:

$\lambda = 0$ in elastic region,

$\lambda = 3\mu_0\tau'_{ij}\dot{e}_{ij}H(\tau'_{ij}\dot{e}_{ij})/Y^2$ in the region of plastic flow.

The model used in this work generalizes the Prandtl-Reiss elastoplastic flow model along with the Mises plasticity criterion, as well as accounts for the anisotropy of plastic deformation (for $\Gamma \neq 0$), the formation of microdamages in rarefaction waves, and thermal effects [5]. It is supposed that the flow limit Y , modules k_0 and μ_0 depend on the temperature, pressure and other state parameters (Steinberg-Guinan model) [7]:

$$Y = Y_0(1 + \beta\varepsilon_u^p)^n(1 - b\sigma(\frac{\rho_0}{\rho})^{1/3} - h(T - T_0)),$$

$$Y_0(1 + \beta\varepsilon_u^p)^n \leq Y_{\max}, \quad Y_0 = 0 \quad \text{at} \quad T > T_m, \quad (3)$$

$$T_m = T_{m0}(\frac{\rho_0}{\rho})^{2/3} \exp(2\gamma_0(1 - \frac{\rho_0}{\rho})),$$

$$\mu_0 = \mu_{00}(1 - b\sigma(\frac{\rho_0}{\rho})^{1/3} - h(T - T_0)),$$

where $\varepsilon_u^p = \sqrt{2\varepsilon_{ij}^p\varepsilon_{ij}^p/3}$ – is the plastic deformation tensor intensity; T_m – melting temperature of the material; $Y_0, Y_{\max}, T_{m0}, \beta, b, \gamma_0, \mu_{00}$ – material constants. It is considered that $\sigma_* = \sigma_*^0 Y/Y^0, \sigma_*^0$ is a material constant. Numerical modeling of impulsive impacts gives a possibility to neglect strain anisotropy of the materials (in case of adiabatic flow) and to assume $\text{div} q = 0$ and $\Gamma = 0$ in equation (1). Thus from equation (1) it follows that:

$$s_{ij}^{\nabla} + \lambda s_{ij} = 2\mu \dot{\varepsilon}_{ij}, \quad s_{ij}s_{ij} \leq \frac{2}{3}Y^2. \quad (4)$$

Equations (4) written for the deviator components of strain tensor are amplified with state equation for the spherical part of strain tensor $\sigma = -p$ (p – pressure):

$$p = p(\rho, U), \quad (5)$$

here U – specific internal energy. State equation is considered in Mie-Gruneisen type [2,7] written in the form:

$$p = l_1(1 - \frac{\rho_0}{\rho}) + l_2(1 - \frac{\rho_0}{\rho})^2 + l_3(1 - \frac{\rho_0}{\rho})^3 + \gamma_0 \rho_0 U, \quad (6)$$

where l_1, l_2, l_3, γ_0 are material constants that are known for the wide class of materials used in calculations. The whole variety of grounds with different mechanical and physical properties is presented as porous multicomponent medium consisting of solid particles, liquid (water) and gas (air). Characteristics of different types of grounds depend on structure, shape and location of solid particles, percentage of gas and liquid. Following Lyakhov, a ground is called non water-saturated (or air-dry grounds) [3] if air content by volume is much higher than the water content in it. Otherwise the ground is called water-saturated. Thus the ground may be considered as three-component medium (solid particles, water and air) and its characteristics depend on the volume content of each component which can vary over a wide range. Let d_i denote respectively a volume content of air ($i = 1$), water ($i = 2$), and solid ($i = 3$) components in the ground. These values are related: $d_1 + d_2 + d_3 = 1$. If ρ_0 is an initial density and p_0 is an initial pressure

then

$$\rho_0 = \sum_{i=1}^3 d_i \rho_i.$$

On this assumption, following Lyakhov, we have ground's state equation [3]:

$$\frac{\rho_0}{\rho} = \sum_{i=1}^3 d_i \left[\frac{k_i(p - p_0)}{\rho_i c_i^2} + 1 \right]^{-\frac{1}{k_i}}. \quad (7)$$

Here k_i – isentropic exponents, c_i – sound velocities in these components for the case of initial pressure (atmospheric pressure).

In actual numerical calculations instead of formula (7) it is more convenient to use the dependence of pressure on density in an explicit form. This dependence can be approximated by a cubic polynomial with respect to compression $\mu = \frac{\rho_0}{\rho} - 1$:

$$p = a_0 + a_1\mu + a_2\mu^2 + a_3\mu^3. \quad (8)$$

Polynomial coefficients are determined on account on (7) using Newton interpolation polynomial or the method of least squares.

3 Numerical results

Initial values of other parameters are as follows: air – $k_1 = 1,4$, $\rho_1 = 12 * 10^4$ gr/sm³, $c_1 = 300$ m/sec; water – $k_2 = 3,0$, $\rho_2 = 1,0$ gr/sm³, $c_2 = 1500$ m/sec; quartz – $k_3 = 3,0$, $\rho_3 = 2,65$ gr/sm³, $c_3 = 4500$ m/sec. At small values of pressure ground's compressibility depends on compressibility of air which significantly exceeds the compressibility of water and quartz. At higher values of pressure ground's compressibility depends on compressibility of liquid and solid components. The discrepancy between the data obtained from expressions (7) and (8) is equal to less than 10 percent.

Consider a water-saturated ground with the following composition: 10 percent of air, 32 percent of water and 58 percent of quartz. Various numerical experiments were conducted to study the behavior of shells in the ground which are exposed to intense dynamic loading. These

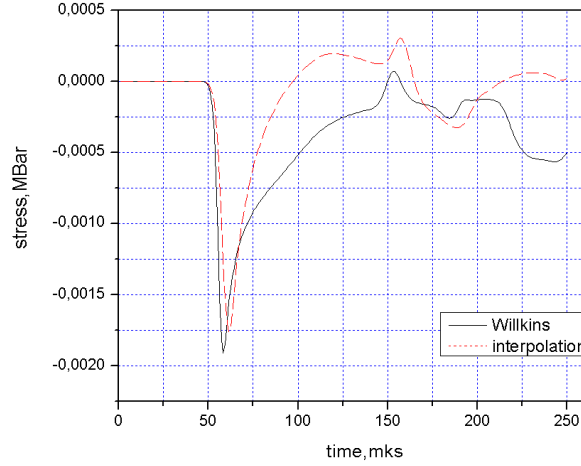


Figure 1. The strain corresponding to different state equations.

studies include an approximation of equations (1)-(6) using a finite-difference scheme of the second order of accuracy that is a development of Wilkins' scheme [2]. Relation (8) is taken as a ground's state equation. It approximates the equation of state with a cubic polynomial. Equation of state (8) is tested for the various cases of a-priori known state equations of medium. Let's consider the case of water and calculate the stress dynamics at a certain point on the computational domain using state equation in Mie-Gruneisen form (solid line in Fig.1), and equation (8) (dotted line in Fig. 1). The expansion of detonation products, shell loading and other physical processes are modeled as well. Some results are presented in the Figures 2–5. The computational domain is a two-dimensional rectangle 21×4 cm. A water-filled steel shell, 0,2 cm thick, is situated in this domain (double fat black lines in the Fig. 2 and Fig. 4). Its dimensions are 16×7 cm. The shell is surrounded with a ground consisting of 10 percent air, 30 percent water and 60 percent quartz. The explosive charge TNT ($0,3 \times 0,1$ cm) closely fits the outside of the envelope (deflected contour in the Fig. 2 and Fig. 4). Certain points are selected to monitor the behavior of the elastoplastic medium parameters. These points are called indicators

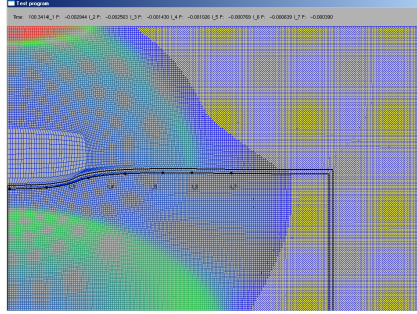


Figure 2. Stress state, $t=100$ mks

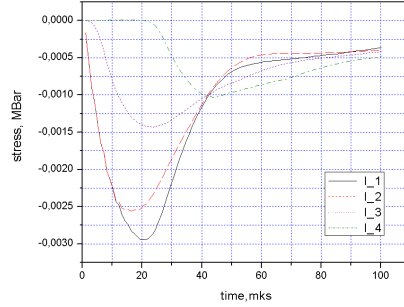


Figure 3. Stress distribution.

and are marked as follows: I-1, I-2, \dots , I-7. They are situated in the inside of the shell.

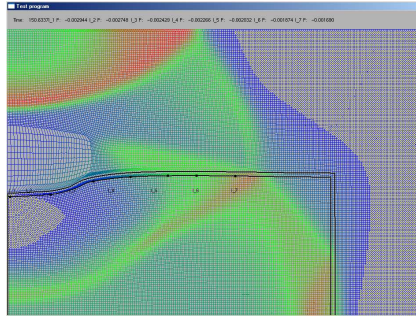


Figure 4. Stress state, $t=150$ mks

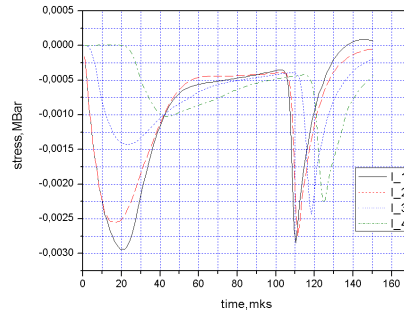


Figure 5. Stress distribution.

Qualitative change in the physical state of the shell and the medium inside and outside it over the time is displayed in Fig. 2 and Fig. 4 at the $t = 100$ mks and $t = 150$ mks correspondingly. Figure 2 displays a wave pattern and the shell's state at $t = 100$ mks in two dimensions. Stress dynamics for the early stages of the loading process up to the $t = 100$ mks is displayed in the Fig. 3. Indicators I-1, I-2 are located directly under the explosive charge (Fig. 2). Thus experimental results for these indicators (Fig. 3) are close enough and have more pronounced

wave profile compared to the indicators I-3, I-4 located farther. The stress state of the shell at the $t = 150$ mks is displayed in the Fig. 4. It is easy to observe that the shell exposed to an explosive loading is deformed and the wave reflected from hard walls of the computational domain. Stress distribution for the later stages of the loading process up to time $t = 150$ mks is displayed in the Fig. 5. One may observe a new surge at the $t = 110$ mks caused by the reflected wave.

4 Conclusions

A new mathematical model was developed. A stress-strain state of underground storage tanks, exposed to intense dynamic loads and filled in with water or other materials was simulated. The ground was considered as a three-component medium (solid particles, water and air) and its characteristics depended on the volume content of each component which can vary over a wide range. State equation for the ground is approximated by a cubic polynomial with respect to the degree of compression. Numerical results illustrate the evolution of the coupled problem, namely the interaction of ground and elastoplastic shell under explosive loading. This work is supported by STCU (grant 4624) project.

References

- [1] Baum F.A., Orlenko L.P., Staniukovich K.P., et al., *Physics of Explosion*, [in Russian] Nauka, Moscow (1975) pp. 704.
- [2] Wilkins M.L. *Modeling the behavior of materials*. Struct. Impact and Grashworth. Proceeding of International Conference.V.2, London, New York, 1984, pp. 243–277.
- [3] Lyakhov G.M., Pokrovskii G.I., *Blast Waves in Soils*, [in Russian] Gosgortekhnizdat (1962), pp. 99.
- [4] Rybakin B. *Computer Modeling of Dynamic Processes*. CSJM, v.8, N 2(23), 2000, pp.150–180.

- [5] Kiselev A.B., Yumashev M.V. *Deforming and fracture under impact loading. The model of thermoelastoplastic medium*. J. Appl. Mech. Tech. Phys., vol. 31 no. 5, 1990, pp. 116–123.
- [6] Lugovoi P.Z., Meish V. F., Rybakin B. P., Secieru G. V. *Numerical simulation of the dynamics of a reinforced shell subject to nonstationary load*. In: Springer, International Applied Mechanics, 2008. Vol.44. No 7, pp. 788–793.
- [7] Lugovoi P., Meish V., Rybakin B., Secieru G. *The numerical investigation of reinforced shell undergo dynamic load*. in: Annual symp. Of the Institute of Solid Mechanics of Romanian Academy (SISOM), May 28-29, Bucharest (2009), pp. 125–129.
- [8] Rybakin B., Russeva E., Secieru G. *Numerical investigation of the process of detonation waves interaction with an elastoplastic target*. Buletinul ASM, Matematica, N.2(42), 2003, pp. 113-122.
- [9] Rybakin B. *Numerical methods for multi-processor computers*. Chisinau: USM, CECMI, 2008. pp. 348. [in Russian]

Elena Gutuleac

Received January 9, 2012

Elena Gutuleac

Institution: Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova

Address: Chisinau, str. Academiei 5, of.301

Phone: 73-81-05

E-mail: russevaelena@gmail.com

Parsing the Dictionary of Modern Literary Russian Language with the Method of SCD Configurations. The Lexicographic Modeling

Neculai Curteanu, Svetlana Cojocaru, Eugenia Burcă

Abstract

This paper extends the experience of parsing other five, sensibly different, Romanian, French, and German largest dictionaries, to **DMLRL** (Dictionary of Modern Literary Russian Language) [18], using the optimal and portable parsing method of SCD (Segmentation-Cohesion-Dependency) configurations [7], [11], [15]. The purpose of the present paper is to elaborate the lexicographic modeling of **DMLRL**, which necessarily precedes the sense tree parsing dictionary entries. The following *three* SCD configurations are described: the *first one* has to separate the lexicographic segments in a **DMLRL** entry, the *second* SCD-configuration concentrates on the SCD marker classes and their hypergraph hierarchy for **DMLRL** primary and secondary senses, while the *third* SCD configuration hands down the same modeling process to the atomic sense definitions and their examples-to-definitions. The dependency hypergraph of the third SCD configuration, interconnected to the one of the second SCD configuration, is specified completely at the atomic sense level for the first time, exceeding the SCD configuration modeling for other five dictionaries [15], [14]. Numerous examples from **DMLRL** and comparison to **DLR-DAR** Romanian thesaurus-dictionary support the proposed **DMLRL** lexicographic modeling.

Keywords: new approach to dictionary entry parsing; the parsing method of SCD configurations; parsing the largest Romanian, German, French, and Russian dictionaries; lexicographic modeling.

1 Dictionary Entry Parsing with SCD Configurations

The general idea behind parsing a thesaurus or dictionary can be reduced to transforming a raw text entry into an indexable linguistic resource. The typical representation of the parsing result of a dictionary entry is its *sense tree structure*.

The aim of this paper is to prepare the **DMLRL** (Dictionary of Modern Literary Russian Language) [18] for parsing with the *method* of SCD (Segmentation-Cohesion-Dependency) *configurations* [7], [15], starting with its necessary *lexicographic modeling* [16], [1]. We rely on the experience of modeling and parsing very efficiently other *five* largest, complex, and sensibly different thesaurus-dictionaries [11], [15]: **DLR** (The Romanian Thesaurus – new format) [2], [11], **DAR** (The Romanian Thesaurus – old format) [28], **TLF** (Le Trésor de la Langue Française) [23], **DWB** (Deutsches Wörterbuch – GRIMM) [17], **GWB** (Göthe-Wörterbuch) [17].

An SCD *configuration* has the following computational components [7], [15], [27]: • A set of *marker classes*: a *marker* is a boundary for a specific linguistic category; • A *hypergraph-like hierarchy* that establishes the dependencies among the marker classes; • A *searching (parsing) algorithm*. The parsing algorithm is designed to perform the following actions: recognize the markers within the text, identify the text structures / spans they bound, and classify these structures according to the pre-assigned hierarchy of marker classes. The algorithm is applied to a *specific* SCD configuration of marker classes and hierarchy, strictly depending on the *semantics* standing behind *that* SCD configuration. Such a semantics involves specific markers, marked categories, and their (partial ordering) hierarchies to be applied along the corresponding text span (or lexicographic segment) to be parsed.

The developed parsing strategy merges the following three SCD configurations: *the first one* has to separate the main *lexicographic segments* [22 :2] of a thesaurus entry; *the second* SCD-configuration should parse each *entry segment*, concentrating on the sense description segment and its sense-tree extraction [11], [12], [14], [15], [16]. This partic-

ularly important SCD-configuration, which obtains the entry *sense tree* exclusively from the sense *marker sequences*, coincides with the DSSD algorithm in [11]. This algorithm (and SCD-configuration) continues with a *third* SCD-configuration that parses each node in the generated sense-tree for obtaining the atomic definitions / senses (*i.e.* finest-grained meanings) of the entry, according to the lexical-semantics modeling of the thesaurus-dictionary; *e.g.* for **DLR-DAR** [11], [14], [15], for **DMLRL** [16], [1], and also the remarks concerning the new types of definitions / senses, definition examples, etc.), *i.e.* their lexicographic types and dependencies.

Parsing with SCD configurations means a good (sometimes, thorough) measure of prerequisite semantic modeling of the text, establishing of the marker classes for syntactic-discursive structures driven by certain precise semantics, determining the dependency hypergraph(s) for the considered marker classes, recognition of the markers in the text, and extraction of the *marker sequences* (only). In such a concrete SCD configuration, parsing means to compute the dependency relations between (among) the markers in the marker sequences of the text, according to the dependencies incorporated *ab initio* in the pre-established *dependency hypergraphs* for the *marker classes* of the configuration.

To notice the important computational characteristic that parsing with SCD configurations is a *procedural*-oriented tool and a completely *formal grammar-free* one, the latter device being proved to be cumbersome and inefficient when applied to free, general, or specialized (such as dictionary entry) kinds of natural language texts.

2 Homonymic Entries in DMLRL

The homonymic entries in **DMLRL** (**D**ictionary of **M**odern **L**iterary **R**ussian **L**anguage) are discriminated by indexing each of the homonyms with Arabic numerals followed by dot, all in *Arial font*, *Regular* and *Bold* format. These indexes are positioned in front of each homonym-word lemma, enumerating increasingly all the homonyms of the same word-lemma in the dictionary. An example of *four* homonymic entries

of the word "БЫЧОК" is present in DMLRL [18 :860-861].

The first two of these entries may cause the same possible ambiguity between the second sense of "БЫЧОК" *first entry*, introduced by the sense marker "2." (font Arial; correct font: Times New Roman), and the index of the second homonymic entry, starting with the similar but slightly different marker "2." (Times New Roman; correct font: Arial). If the parsing program works properly and associates unequivocally the homonymy index to the DMLRL entry lemma (which is written with bolded capital Cyrillic letters), then there should not appear ambiguities when discriminating and parsing the lexically independent homonymic entries in DMLRL. The (ambiguity-introducing) original example is [18 :861]:

1. БЫЧОК, чка, м. **1.** Разг. Уменьш.-ласк. к бык (1. Бык в 1 знач.); молодой бык. В стайке у Кузнецовых рос бычок, низколобий, красный, с рожками, похожими на шишки. Задорн. Амур-Батюшка. Лоси сбрасывают рога; старые самцы в декабре — январе, молодые бычки — в конце февраля — в марте. Формоз. Спутн. Следопыта. ◇ В сравн. [Сергей] выслушивал предложения молча, насупившись, склонив, как бычок, голову и напрягив шею. Первенц. Дир. Томилин. ◇ Смотреть, глядеть и т. п. бычком. Смотреть хмуро, исподлобья.— Ну, а парнишку-то!.. сажай и его! Что, смотрю, он у тебя таким бычком глядит, слова не скажет. Григор. Рыбаки. Лешка смотрел на него [мир] не как прежде — широко открытыми, ясными серыми глазами, — а бычком, исподлобья и ожидал от него одних неприятностей. Дубов, Горе одному. ◇ Пить бычком. См. Пить.~ **Сказка про белого бычка.** См. Сказка.

2. Перен. Разг. О молодом упрямом человеке (обычно в функции сказуемого).— Эх ты, бычок несуразный .. грохотал Сиволап шатающемуся Кромулину. Леон. Конец мелк. чел. Всех широковцев обозвал он кротами, а Яшку — бычком, бездельником-буяном. Панфер. Бруски. ~ **Быть бычку на веревочке.** См. Быть.

— Поликарпов, 1704: быччк; Нордстет, 1780: бычок.

2. БЫЧОК, чка, м. **1.** Рыба отряда окунеобразных.

Черноморские, каспийские бычки. Я и механик удили с палубы рыбу и нам попадались очень крупные, толстоголовые бычки. Чех. Остр. Сахалин. Они поймали .. одну горбушу и двух бычков-подкаменщиков с пестрой окраской и оранжевой каймой на темно-оливковом спинном плавнике. Арсен. Дерсу Узала. До чего ж прозрачна байкаль

3 The Main Lexicographic Segments in DMLRL

In [11], [12], [14], [15] there have been analyzed the first and second SCD configurations of the following five thesaurus-dictionaries: **DLR** (The Romanian Thesaurus – new format) [2], [11], **DAR** (The Romanian Thesaurus – old format) [28], [13], [15], **TLF** (Le Trésor de la Langue Française) [23], [15], **DWB** (Deutsches Wörterbuch – GRIMM) [17], [15], and **GWB** (Göthe-Wörterbuch) [17], [15]. This knowhow is applied and extended in this paper to **DMLRL** (Dictionary of Modern Literary Russian Language) [18].

The *first SCD configuration* has to recognize the lexicographic segments of a **DMLRL** entry. **DMLRL** comprises (at least) five types of lexicographic packages / segments: **(1)** a *morpho-lexical* package / segment, **(2)** the *sense description* segment, **(3)** a *TildaDef* package or segment of definitions (see subsection 3.3), **(4)** the *morpho-syntactic variant* segment, and **(5)** the *etymology* segment of the word-lemma. The morpho-lexical definition package is obligatorily present at the beginning of each entry, immediately after the word-lemma. The morpho-lexical package may occur also at the sense lower-levels of the entry sense tree. The *TildaDef* package can be attributed not only to any (sub)sense description level of the entry but also to the root-sense (zero-level sense hierarchy). When this package / segment begins at *new paragraph* (*NewPrg* typographic marker), the *TildaDef* package is assigned to the entry root-sense. The same *NewPrg* lexicographic marker is met in **DAR** thesaurus-dictionary [15], [14] (see subsections 3.2, 5.2).

In the following subsections, some examples of **DMLRL** lexicographic segments are given. We notice that the structure of lexico-

graphic segments for large thesaurus-dictionaries, recognized within the *first SCD configuration*, is linear and simple, in general [14], [15]. However, remarkable exceptions are the oldest dictionaries studied, namely the German **DWB** [17] and the Romanian **DAR** [28], [15], whose design began in 19-th century for both.

3.1 The Morpho-Lexical Segment of a DMLRL Entry

The entry **БЫТЬ** is enlightening for the morpho-lexical segment: this lexicographic package / segment covers the first rows, from the word-lemma until the first primary sense introduced by the marker "I.", in bold [18 :856]. With this marker begins the most important lexicographic segment of **DMLRL** entries (and in any dictionary), *viz.* the segment containing the lexical-semantics descriptions of entry senses, called the *sense description segment*.

БЫТЬ, *наст. не употр. кроме 3 л. ед. е с т ь и (устар.) 3 л. мн. с у т ь, буд. б у д у, б у д е ш ь, прош. б ы л, б ы л а, б ы л о (с отрицанием: н е б ы л, н е б ы л а, н е б ы л о), повел. б у д ь (т е), прич. действ. прош. б ы в ш и й, деепр. б у д у ч и, несов., неперех.*

I. Как самостоятельный глагол означает: **1.** Существовать. *Не говори с тоской: их нет; Но с благодарностью: были.* Жук. Воспоминание. *В дымном зареве вставал рассвет. Город был. Сегодня нет его.* Сурк. Город О. Прошлое человечества — драгоценная сокровищница неисчислимых богатств .. Эти богатства были, то есть существовали когда-то реально во времени. Шагинян, Воскреш. из мертвых. ◇ Жил-б ы л, жила-б ы л а и т. п. жил да б ы л. Нар.-поэт. Употр. как начало повествования, сказки и т. п. Жила-была вдова, Тому лет восемь, бедная старушка. Пушкин. Домик в Коломне. Во время оно жил да был В Москве боярин Михаил, Прозваньем Орша. Лерм. Боярин Орша. ◇ О каком-л. времени, периоде, поре и т. п. Была осень. Был полдень. Была та смутная пора, Когда Россия молодая, В бореньях силы напрягая, Мужала с гением Петра. Пушкин. Полтава.

Other examples of **DMLRL** morpho-lexical segments are the following (shaded part) ones [18 :781]:

БРОШЮРОВАТЬ, р у ю, р у е ш ь, *прич. страд, прош.*
б р о ш ю р о в а н н ы й, а я, о е, *несов. перех.* Сшивать, скреплять
отпечатанные листы в книгу или брошюру соответственно нумера-
ции. При издании журнала мне доверялось только брошюровать
тираж, написанный под копирку в несколько экземпляров. А.
Гусев, От Эльбр. до Антаркт.

— С иным (*устар.*) напис. и произнош.: б р о ш и р о в а т ь.—
Даль: брошировать; Ушаков, 1934: брошюровать.— От франц.
brocher — сшивать листы книги.

БРОСАТЬ, а ю, а е ш ь, *несов.*; **бросить**, б р о ш у, б р о с и ш ь,
прич. страд, прош. б р о ш е н н ы й, а я, о е, *сов.; перех. и неперех.*
1. Перех. Резким движением, взмахом заставлять перемещаться
в воздухе в каком-л. направлении копн, что-л.; кидать (в 1 знач.).
Бросить камень, палку.

Within *MorfDefs* of the morpho-lexical segment, several *SpecDefs*,
SpSpecDefs, *LexVarDefs* etc. may be inserted (see subsection 5.2).

3.2 The Sense-Description Segment

The investigation of the lexicographic segment devoted to the lexical-
semantics sense description is focused in Sections 4 and 5 below, which
contain interesting examples for sense and definition description mark-
ers, together with their dependencies, represented as procedural, inter-
connected hypergraphs. The complete analysis of this segment consti-
tutes the *second* and *third SCD configurations*, which is naturally the
most important enterprise for the lexicographic modeling of the dictio-
nary entry parsing process.

3.3 The *TildaDef* Package / Segment of Definitions

The *TildaDef* definition is introduced by the **DMLRL**-specific marker
tilda "˜", being written in *bolded* and *italics* Times New Roman font,

at the end of a sense / subsense description. The *TildaDef* package is *NewPrg* non-marked when attached to an entry having just the sense-root or to the proper subsenses of the word-lemma, but *NewPrg* marked when assigned to the root-sense of an entry with proper subsenses.

In the present **DMLRL** lexicographic modeling associated to the parsing method of SCD configurations, the *TildaDef* definition header is defined as the *bold* and *italic* text span situated between the *Tilda* "˘" marker and: (1) a *literal enumeration* marker, (2) the first *RegDef*, or (3) the first *RefDef* (*Reference Definition*) part that follows (see Fig. 2). Samples of *TildaDef* headers (see the example below): **Бросать/бросить деньги на ветер.**; **Бросать оружие.**; ... **Будет и на нашей, моей и т. п. улице праздник.**

In general, the *TildaDef* package of definitions is attached to a proper subsense (at least one level below the root-sense level) of the dictionary entry, but it is also possible that *TildaDef* (package) to be the single definition of the entry root-sense. When in final position, the *TildaDef* package is actually assigned to the *root-sense* of the entry, usually also *NewPrg* marked; in this situation, it may be considered as a *special* lexicographic *segment* of that **DMLRL** entry.

For instance, the *TildaDef* definition package for the entry **БЫТЬ** spans on almost two pages of the **DMLRL** thesaurus-dictionary. It ends up with etymological description segment of the entry (see previous subsection), thus the *TildaDef* package assigned to the entry **БЫТЬ** can be assimilated to a root-sense segment of the entry. Here there are examples of *TildaDefs* associated both to proper subsenses, as in **БРОСАТЬ** [18 :772]:

... .. ◇ Б р о с а т ь напрасно, зря; б р о с а т ь направо и налево. — *Весь разговор из-за каких-то десяти-двадцати лир .. — Зачем напрасно бросать двадцать лир!* Верес. Паутина. *Деньги не залеживались у него: он бросал их направо и налево, не отказывая никому.* Пермитин, Перв. любовь. // *Переставать пользоваться чем-л. как ненужным, бесполезным. Повелено было всем генералам и офицерам уменьшить, по возможности, свои экипажи и жечь все ими бросаемое.* Пушкин. Зап. Моро де-Бразе. *Последний уцелевший немецкий танк был бесславно брошен*

экипажем. Левченко, Капли воен. грозы. ~ **Бросать/бросить** **деньги на ветер**. См. Д е н ь г и. **Бросать оружие**. Сдаваться, отказываться воевать. [Суконщики] *последние бросили оружие и уступили превосходной силе*. Пушк. Ист. Пугачева. **Оторви да брось**. См. О т р ы в а т ь. **Поднять да бросить**. См. П о д н и м а т ь. **Хоть брось**. О ком-, чем-л. плохом, никуда не годном, не поддающемся исправлению и т. п. [Хлестова:] *А ты, мой батюшка, не исцелим, хоть брось*. Изволил вовремя явиться! Гриб. Горе от ума.— *Ну, что там? — Ось сломалась. — Барин для порядка ее потрогал. — Да, хоть брось*. Тург. Помещик.

4. *Перех.* Класть что-л. небрежно, не на свое место; кидать (в 3 знач.).— *Ты, Оксана, смотри, не бросай так ключа от твоей комнаты; не пропало бы что у тебя*. Данил. Беглые в Новороссии
... ..

or to the root-sense, as in **БЫТЬ** [18 :857]:

5. *Разг.* Употр. в формах будущего времени в знач. связи настоящего времени. ◇ При выяснении происхождения, родства, имени, социального положения и т. п.— *Гостит у нас .. Иван Иванович Мизинчиков, тебе будет двоюродный брат, кажется*. Дост. Село Степанчиково.. — *Ты сам откуда же будешь? — Мы рязанские*. Сераф. В пути.— *Тетя, — окликает ее [женщину] Вика храбро, — вы будете художница?* Лидина, Леванти.

~ **Будет и на нашей, моей** и т. п. **улице праздник**. См. П р а з д н и к. **Будь здоров**. См. З д о р о в ы й. **Будь не во гнев; не во гнев будь сказано**. См. Г н е в. **Будь спокоен, будьте спокойны**. См. С п о к о й н ы й. **Будь то..., или...; будет ли то..., или...** Употр. для выражения предположения при перечислении, сопоставлении и т. п. чего-л. *Начиная работать над каким-нибудь портретом, будь то изображение самое известное и документальное, или, наоборот, потерявшее свое имя, всегда можно ожидать любых осложнений*. Немилова, Загад.стар.картин. *Но всякая материальная сила, будь то сила мужского тела или же сила машины, нуждается еще в духовном водительстве*. Горыш.

Водопад. **Будь (ты, он, она) (трижды) проклят, проклята.**
См. П р о к л и н а т ь.

The lexicographic *Tilda* package / segment illustrated above contains:

(a) Several *TildaDef* headers, namely: “*Будет и на нашей, моей и т. п. улице праздник.*”; “*Будь здоров.*”; “*Будь не во гнев; не во гнев будь сказано.*”; “*Будь спокоен, будьте спокойны.*”; “*Будь то..., или...; будет ли то..., или...*”;

(b) Several *RefDefs* (*Reference Definitions*), namely: “См. П р а з д н и к.”; “См. З д о р о в ы й.”; “См. Г н е в.”; etc., completing the *TildaDefs*;

(c) A general form of the *TildaDef* shape, made up of a *TildaDef* header, followed by a *RegDef* and two *DefExems* (quoted text and its *sigle* – i.e. its bibliographic source reference(s); the term *sigle* is assumed from French): “*Будь то..., или...; будет ли то..., или...* Употр. для выражения предположения при перечислении, сопоставлении и т. п. чего-л. Начиная работать над каким-нибудь портретом, будь то изображение самое известное и документальное, или, наоборот, потерявшее свое имя, всегда можно ожидать любых осложнений. Немилова, Загад. стар. картин. Но всякая материальная сила, будь то сила мужского тела или же сила машины, нуждается еще в духовном водительстве. Горыш. Водопад...”

(d) Other *TildaDefs* of the package; see the hypergraph in Fig. 2, showing the (sequences and) dependencies for the atomic definitions and examples-to-definitions within the *lexical-semantics projection* of the primary and secondary senses into atomic senses of **DMLRL**.

A *special situation*, interesting from several points of view, demonstrates the following entry [18]:

АВГИЕВЫ. ~ **Авгиевы конюшни** (чего-л.). а) Об очень загроможденном, захламленном месте, помещении. *Письменный стол наш представляет авгиевы конюшни, и только теперь я мог обрести клочок бумаги.* Мусорг. Письмо В. В. Стасову, 31 марта 1972. б) О чем-л. находящемся в крайне запущенном состоянии;

о беспорядке, неразберихе где-л. — Говорят, ревизор энергически принялся за очистку авгиевых конюшен попечительства над училищем. Гл. Усп. Бог грехам терпит.

This *TildaDef* package can express, *by itself* (even with a single component definition), the lexical-semantics sense contents of **DMLRL** entry. The entry **АВГИЕВЫ** above, whose sense is defined basically through a *TildaDef* definition header, is further refined by *literal enumeration*. A similar type of entry sense definition can also be met in **DLR-DAR**, where entries can be described exclusively through a *BoldDef* or *ItalDef* definition [10], [11], [12], [13]. Equally, the literal enumeration may refine such zero-level definitions of atomic kind. Thus, in case of **DMLRL** dictionary, *RegDef*, *TildaDef*, and *RefDef* are *autonomous definitions*, in the sense described in subsection 5.2, initially proposed for **DLR-DAR** dictionaries [12], [14]. Section 5 comes into details on the atomic definitions / senses and types of examples-to-definitions.

3.4 The Morpho-Syntactic Variant Segment

The *Morpho-Syntactic Variant Segment* describes an independent subentry, associated as a syntactic variant to the basic entry. Typical examples are *adverbial* forms (**По-бычьи**) associated to certain *adjectives* (**БЫЧИЙ**), as in the following sample [18 :860, 772]. The *Morpho-Syntactic Variant Segment* is located between the *sense description* segment, possibly ended with a *Tilda package* (or segment), and the *etymology* segment of **DMLRL** entry.

БЫЧАЧИЙ, ья, ье. *Разг.* 1. Относящ. к быку, быкам (1. Бык в 1 знач.),...

2. Свойственный быку; такой, как у быка.

По-бычачьи, нареч. То же, что по-бычьи. *Гараська по бычачьи мотнул головой.* Аник. Гараська-диктатор.

— Слов. XI—XVII вв.: бычатий; Вейсманн, 1731, с. 454: бычачий; Росс. Целлариус 1771,

БЫЧИЙ, ь е, ь я. 1. Относящ. к быку, быкам (1. Бык в 1 знач.),

3. В составных народных названиях растений. *Бычья трава. Бычий ноготок.*

По-бычи, нареч. Как бык (1. Бык в 1 знач.), подобно быку. *Старший сержант по... ..*

— Срезневский: б ы ч и й; Вейсманн, 1731, с. 609: б ы ч и й; Нордстет, 1780: б ы ч и й; БАС 1948: п о-б ы ч ь и.

3.5 The Etymology Segment

The etymological description segment (shaded part), which always ends a **DMLRL** entry, is illustrated here also on the entry **БЫТЬ** [18 :856]. The **DMLRL** etymology segment is always introduced by the specific *etymological-dash*, *NewPrg* (*New Paragraph*) marked, and written with (two points) smaller font than the (Times New Roman) common text font measure of **DMLRL** dictionary entries.

... .. *Чтоб тебе, ему и т. п. пусто было.* См. П у с т о . *Чтобы духу твоего, вашего и т. п. не было.* См. Д у х . *Чтобы неповадно было.* См. Н е п о в а д н о . (Я) н е я б у д у , е с л и н е... Употр. для выражения твердой уверенности или решительного намерения.— А где ж она, родительница-то? али спряталась? Не я б у д у , е с л и н е сидит где-нибудь там, за ширмами. Дост. Село Степанчиково.. — Я не я б у д у , е с л и н е окажется [в рапорте], что мы вырвались после отчаянной борьбы против охраны. Мстислав. Грач — птица весенняя.

— В иной (разг.) форме: *деепр. б ы в ш и*; в иной (устар.) форме: *деепр. б ы в*.— Срезневский: б ы т и; Берында, 1627: б ы с т ь; Вейсманн, 1731, с. 351: да б ы т ь так, не б у д е т , не б ы т ь удаче; Лекс. 1762: б ы т ь.

4 The Dependency Hypergraph at Sense Marker Classes in DMLRL

4.1 Primary and Secondary Senses in DMLRL. Examples, Dependencies

The *primary sense* markers in **DMLRL** pointed out so far by the lexicographic analysis are: **(1)** capital Roman numerals followed by a dot (**I.**, **II.**, **III.**,...etc.), in bold (*LatCapNumb_Mark*), and **(2)** Arabic numerals followed by a dot (**1.**, **2.**, **3.**,... etc.), in bold (*ArabNumb_Mark*). The markers of these classes are positioned at the beginning of the text row, in fact, at new paragraph (*NewPrg* marker), except for the *first sense markers* (**I.**, **1.**), which usually does not occur at new paragraph.

The sense markers of the class denoting Roman capital numerals followed by a dot (**I.**, **II.**, **III.**,...etc. or simply, *LatCapLett_Enum*) represent the top of the sense hierarchy in **DMLRL**. These markers establish the lexicographic limits for the *most general senses* of the word-lemma. To notice that they are the equivalent of the marker class denoted by bolded Latin capital letters **A.**, **B.**, etc. (abbreviated as *LatCapLett_Enum*) in **DLR** [9], [11], [12].

The sense marker class of *Arabic numerals* followed by dot, point (**1.**, **2.**, **3.**,... etc.), in bold (*ArabNumb_Enum*) stands for the second level of primary sense representation in **DMLRL**. The place of these two sense marker classes is displayed within the hypergraph of Fig. 1 below. The sense marker classes *LatCapNumb_Enum* and *ArabNumb_Enum* are considered to be the set of **DMLRL** *primary senses*, similarly to **DLR-DAR** lexicographic modeling [9], [11], [15].

We placed the *two-oblique-bars* “//” sense marker, which is specific to **DMLRL**, on the *third level* of the hierarchical dependency structure of **DMLRL** senses (Fig. 1). In the same time, the sense marker “//” is considered to be the first element of the two-markers set {//, ◇} denoting the *secondary senses* in **DMLRL**. The sense marked by “//” is in lexical-semantics subordination to (or subsumed by) any other primary sense marked by an element in the marker classes {*LatCapNumb_Enum*, *ArabNumb_Enum*}, when they exist in the entry text.

Otherwise (when a primary super-ordinated sense lacks), the secondary sense marker “//” may occur immediately under the topmost level of the **DMLRL** sense hierarchy. The marker “//” is embodied explicitly into the entry text, even for the case when this level has only one element of this type. For instance [18]:

АБРИКОСОВЫЙ, а я, о е. 1. Относящийся к абрикосу, абрикосам (в 1 знач.). ◇ А б р и к о с о в о е дерево. То же, что абрикос. // Состоящий из абрикосов. *Абрикосовый сад.*

2. Относящ к абрикосу, абрикосам (во знач 2.) *Абрикосовая косточка.* // Приготовленный из абрикосов, с абрикосами. *Абрикосовый сироп. Абрикосовое варенье.*

We notice that *RegDefs* in the // -marked subsenses to the primary senses in the above entry are refined by the so-called *DictExem*, i.e. examples-to-definitions given by the **DMLRL** authors. Usually, *DictExems* are separated from *DefExems* that follows through the **DMLRL**-specific marker *traverse* “□”. See subsection 5.1-(ii) for further discussion on “□” marker, the first sense description “1.” of entry **ВЕДУЩИЙ** that follows (and Fig. 2). In this entry, the secondary sense “//” is refined through literal enumeration. In analogy with the **DLR** hypergraph of sense dependencies, we associate the **DMLRL** “//” marker with the **DLR** “◆” sense marker: they are both secondary sense markers and subsume the similar secondary sense marker denoted in both dictionaries by the *empty-diamond* “◇” (see below) [11], [15], [18].

ВЕДУЩИЙ, а я, е е. 1. Идущий впереди; головной. *Ведущий самолет. Каждый из ведущих броненосцев больше всего осыпался неприятельскими снарядами. Нов.-Прибой, Цусима.* // В знач. сущ. В е д у щ и й, е г о, м, В е д у щ а я, е й, ж. а) Тот, кто ведет, возглавляя какую-л. группу. *В тайге заблудиться легко, если к тому же окажется самонадеянным и не очень опытным ведущим. Ворон. Волев. прием. Последнее время Драченко ходит у нас в качестве разведчика. А теперь думаем посылать его ведущим. Кудреватых, Стр. нашей жизни.* б) Летчик, летящий на головном самолете, направляющий действия своего ведомого. *За*

те немногие минуты, что они провели в воздухе, Петров сумел оценить уверенную и поистине мастерскую манеру полета своего ведущего. Б. Полев. Пов. о наст. чел.

Actually, the second marker in the *secondary sense* marker set used by **DMLRL** is the "horizontal-empty-diamond", which will be replaced in the **DMLRL** entry text, for the ease of graphical representation, lexical-semantic role, and uniformity, with the **DLR-DAR** sense marker "◇", *i.e.* the "vertical-empty-diamond" or, simply, the *empty-diamond* "◇" marker. The lexical-semantics sense defined by the "◇" marker is subsumed (thus subordinated) by the **DMLRL** sense defined with the sense marker "//".

We associate the secondary sense marker set {◆, ◇} in **DLR-DAR** with the corresponding set of markers {/, ◇} in **DMLRL**, relying on the following facts supported by the current stage of our investigation: (a) ◆ subsumes ◇, thus ◆ subordinates ◇ in **DLR-DAR** (actually, these relations refer to the entry senses introduced by these markers) [11]. (b) Similarly, / subsumes ◇, thus / subordinates ◇ in **DMLRL**. (c) The senses introduced by these markers are considered to be *secondary senses*, each pair in its corresponding dictionary, because of the high similarity of their lexical-semantics description refinement (a concept which we called *lexical-semantic granularity* of dictionary entry senses) [11], [14]. (d) Another argument for the proposed relationship is that these sense markers behave likewise when related to the sense refinement technique of *literal enumeration*: both markers in the above pairs of secondary sense markers, for the dictionaries **DLR-DAR** and **DMLRL**, are interleaving with the literal enumeration, recursively calling each other on several (but finite number of) levels. Typical examples are the entry "CAL" in **DAR**, demonstrated in [15], [14], and the entry "БЫ" [18 :844] in subsection 4.3 below. (e) Finally, preserving similar measures of lexical-semantic granularities in the thesaurus-dictionaries **DLR-DAR** and **DMLRL**, the primary senses (**A.**, **B.**, ...; **I.**, **II.**, ...; **1.**, **2.**, ...) in **DLR-DAR** and (**I.**, **II.**, ...; **1.**, **2.**, ...) in **DMLRL** do not interleave with the literal enumeration(s), while the secondary senses, {◆, ◇} in **DLR-DAR** and {/, ◇} in **DMLRL** do, as noticed in (d) above.

If there is no other higher-level sense marker, the "◇" marker may occur immediately below the root-sense of the entry sense tree, as in the following example [18 :781]:

БРОШЮРНЫЙ, ая, ое. Относящ. к брошюре, брошюрам, связанный с их производством. *Брошюрное шитье*. ◇ **Брошюрная литература**. *Устар.* Литература, издаваемая в виде брошюр.

Besides, the sense derived immediately from the *empty-diamond* marker "◇" can be refined by literal enumeration, as in the example "**БРАТЬ**" below, for the subsense no. "**14.**" (the shaded part) [18 :742].

... ..

14. Перех. С некоторыми существительными (с предлогами и без предлогов) обозначает: производить какое-л. действие в соответствии со значением существительного. *Брать на буксир*. ... ◇ **Брать на прицел, на мушку**. Целясь, готовиться к стрельбе. *Завтра они будут ползти по окопам, закладывать мины, брать на мушку фрица*. Эренб. Буря. *Глядит Громак и молвит: — Есть! Заметил вражью точку, Берет тот кустик на прицел, Припав к ружью наводчик*. Твард. Ив. Громак. ◇ **Брать на учет**. **а)** Заносить в списки лиц, входящих в состав чего-л. (обычно в официальной речи). *Лейтенант брал Бондарева на учет*. Родичев, *Стоял старик на обочине*. **б)** Устанавливать наличие, количество кого-, чего-л. [*Двенадцать комсомольцев*] *ушли..брать на учет богатства, которые надо будет вывозить из мест затопления*. Песков, *Счастье перв. тропы*. **в)** Принимать во внимание, учитывать что-л. *Те, кто работал с ним в лаборатории, удивлялись тщательности его экспериментов — он брал на учет все мелочи, исключал возможность малейшей ошибки*. Гранин, Вар. *второй*. ◇ **Брать под вопрос**. См. **Вопрос**.

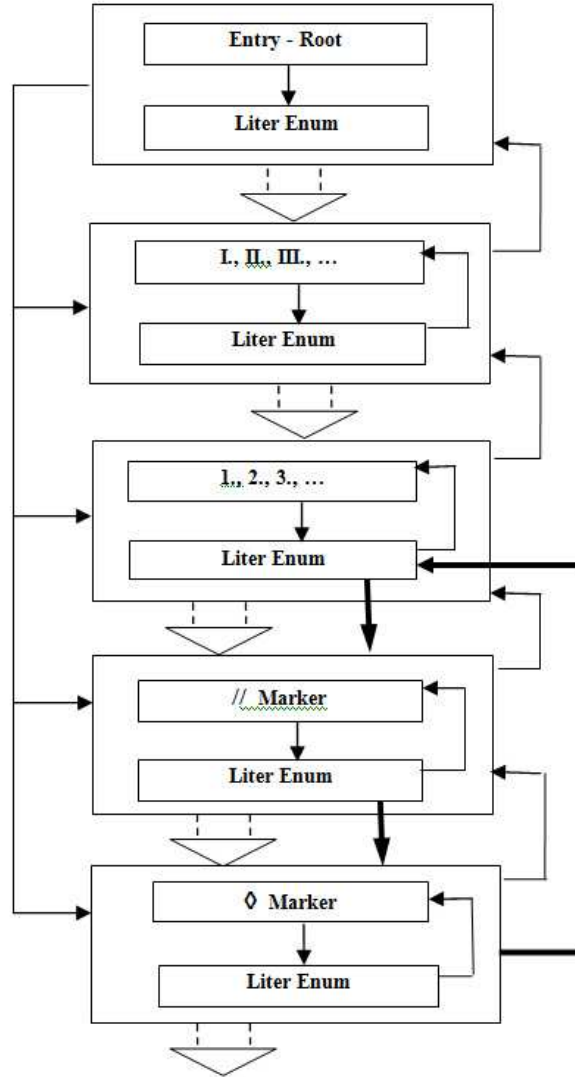


Figure 1. The Dependency Hypergraph at Sense Marker Classes in DMLRL

4.2 The Literal Enumeration and Its Recursive Dependency with DMLRL Sense Markers

The problem of literal enumeration in **DMLRL** is, for the moment, the most challenging one concerning the sense dependencies introduced by **DMLRL** marker classes. This is because one may find entry samples that display a recursion between the *literal enumeration* and the *secondary senses* “//” and “◇” (at least these markers). This level of recursion can be raised towards the higher (primary) senses, or may step down to the atomic senses / definitions. The solution of reducing these recursions to a finite number of cycles, and disambiguation of the cyclic application of secondary sense markers and of the literal enumeration should be consistent with the possible extension of the literal enumeration recursion to the higher or lower levels on the **DMLRL** hypergraph of marker class dependencies, pre-established for **DMLRL** (Fig. 1).

The following lexicographic sense description levels in **DMLRL** are specifiable through literal enumeration:

(1) **Refinement of the primary senses**; for this situation we deliver examples concerning the lexical-semantics refinement introduced by Arabic numerals (*ArabNumb_Enum*), but not the senses marked with Roman numerals (*LatCapNumb_Enum* class). A logical explanation would be that, for the lexical-semantics *granularity measure* of senses introduced by the *LatCapNumb_Enum* markers, the literal enumeration should not be an adequate refinement tool but rather the immediately lower, still primary or secondary levels of sense specification, managed by the *ArabNumb_Enum* marker class, “//” and “◇” markers.

(2) **Refinement of the secondary senses** “//” and “◇” by literal enumeration; for instance, in the entry **ВЕДУЩИЙ**, subsection 4.1.

(3) **Refinement of atomic senses / definitions** by literal enumeration. We have the **DMLRL** entry example of **БЫВШИЙ** [18 :846], where the sense introduced by “◇”, which details the entry root-sense definition, is refined by literal enumeration. Another example is the entry **АВГИЕВЫ** (subsection 3.3 above), whose root-sense is described by a *TildaDef* atomic definition, refined at its turn through

literal enumeration.

4.3 Which Sense Levels Could Refine the Literal Enumeration?

We are interested now in *the reverse situation*: which are the sense levels that could refine the lexical-semantics sense / definition(s) of a *letter* marker (or several, for instance) belonging to the sense refinement procedure of *literal enumeration*? The most interesting case we met (until now) is the entry "БЫ" [18 :844], under the primary sense no. "3." This subsense begins to be refined through literal enumeration, the first sense marker letter "а)" being further detailed with the following sequence of secondary sense markers \diamond , //, \diamond , \diamond . This marker sequence is followed by literal enumeration second letter "б)", further refined by the sequence \diamond , \diamond , \diamond of (secondary) sense markers. The next letter-marker is "в)", with some atomic definitions, followed by the letter marker "г)", which is specified by two secondary subsenses: \diamond , \diamond . The entry excerpt of "БЫ" [18 :844] is illustrative:

... ..

2. В придаточной части сложного предложения обозначает действие, обуславливающее собой то, о чем сообщается в главной части. *Когда б разбойника облавою не взяли, То многие еще бы пострадали.* Михалк. Бешен, пес

3. Обозначает различные оттенки желаемости действия; **а)** Собственно желаемость. *Учился бы сын. Были бы дети здоровы.* \diamond Если бы, когда бы, хоть бы и т. п. О, если бы когда-нибудь *Сбылась поэта сновиденья!* Пушк. Посл. к Юдину. [Николка:] *Хоть бы дивизион наш был скорее готов.* Булгаков, Дни Турб. \diamond С неопр. ф. глаг. *Полететь бы пташечке К синю морю; Убежать бы молодцу в лес дремучий.* Дельв. Пела, пела пташечка.. [Настя:] *Ах, тетенька, голубок! Вот бы поймать!* А. Остр. Не было ни гроша... — *Жара, дедушка Лодыжский .. Нет никакого терпения! Искупаться бы!* Купр. Бел. пудель. // Употр. для выражения опасения по поводу какого-л. нежелательного действия (с отрицанием). *Не заболел бы он.* \diamond С неопр. ф. глаг., имеющей

перед собой отрицание. — *Гляди, — говорю, — бабочка, не кусать бы тебе локтя! Так-таки оно все на мое вышло.* Леск. Воительница. ◇ **Только бы (б) не.** — *По мне жена как хочешь одевайся, .. только б не каждый месяц заказывала себе новые платья, а прежние бросала новешенькие.* Пушкин. Арап Петра Великого. [Варя:] *Не опоздать бы только к поезду.* Чех. Вишневый сад. **б) Пожелание.** **Условие я бы предпочел не подписывать.** Л. Толст. Письмо А. Ф. Марксу, 27 марта 1899. ◇ **С неопр. ф. глаг.** *Поохотиться бы по-настоящему, на коня бы денег добыть, — мечтал старик.* Г. Марков, Строговы. ◇ **В сочетании с предикативными наречиями со знач. долженствования, необходимости, возможности.** [Алеша Бровкин] *сверкнул глазами и понесся .. по гнилым полам приказной избы. Вслед ему косились плешивые понытки: “Потише бы надо, бесстрашной, здесь не конюшня”.* А. Н. Толст. Петр I. ◇ **Только бы (б), лишь бы, Употр. со знач. желательности действия.** [Скалозуб:] *Мне только бы досталось в генералы.* Гриб. Горе от ума. **в) Желание-просьба, совет или предложение (обычно при мест. 2л.).** [Марина:] *И чего засуетился? Сидел бы:* Чех. Дядя Ваня. — *Пошел бы ты к ним счетоводом, полковник.* Павлин. Счастье. — *Ты бы, Сережа, все-таки поговорил с Лидией:* Пришв. Кац. цепь. **г) Желанность целесообразного и полезного действия.** ◇ **С неопр. ф. глаг.** *Вам бы вступить за Павла-то!* — воскликнула мать, вставая. — *Ведь он ради всех пошел.* М. Горький, Мать. ◇ **С неопр. ф. глаг., имеющей перед собой отрицание.** [Лиза:] *А вам, искателям невест, Не нежиться и не зевать бы.* Гриб. Горе от ума.

~ *Во что бы то ни стало.* См. Стать. *Как бы не так.* См. Как. *Кто бы ни был, что бы ни было, как бы то ни было.* См. Быть. *Хоть бы хны.* См. Хоть. *Хоть бы что.* См. Хоть.

— Срезневский: бы; Лекс. 1762: бы.

This example shows that literal enumeration can be further refined through secondary subsenses introduced by the sense markers ”//” and ”◇”. In the previous examples, we have seen that both primary senses (demonstrated for those defined by the marker class *ArabNumb_Enum*,

at this time) and secondary senses can be refined through literal enumeration. We did not (and didn't expect to) find the situation when the literal enumeration to be refined through primary sense marker. Since we have the concrete situation when secondary senses are detailed through literal enumeration, and the reverse holds too (at least for the example above), the two processes are calling each other for a finite (times of) recursion calls that are sequencing the two procedures of lexical-semantics particularization. The problem is to stipulate an *explicit criterion* for stopping effectively the mutual calling of the two refinement processes (through secondary senses and literal enumeration) in a finite number of steps.

Figure 1 provides the scheme of dependencies between the sense marker classes, in **DMLRL**, for the primary and secondary senses, possibly refined through the lexicographic device of *literal enumeration*. The hypergraph of dependencies at the classes of sense markers in **DMLRL** displays in Fig. 1 the *finite recursion* between the blocks of secondary sense markers, // and \diamond , and the literal enumeration: usually, each of the two secondary sense markers may call the literal enumeration, but the **DMLRL** dependency hypergraph specifies that the reverse is also true, *i.e.* the literal enumeration may call, at its turn, each of the secondary sense markers! The *direct calls* made from the marker sub-blocks to the other marker class blocks are put on view with *bolded arrows*.

The procedure, called the "*enumeration closing condition*" for the literal or numeral enumeration, is explained in the sequel. The programming solution for a deterministic condition of a *finite* number of cycles, when mutual calls of (the mentioned) marker classes are performed, is to check the following *closing condition*: for getting out of the (literal or numeral) enumeration (or, in other words, to terminate the enumeration procedure), after the last letter (number) closing the enumeration list, the sense level description is raised at least *one unit higher* than any of the marker levels used as subordinated sense markers under the (literal or numeral) items in the enumeration list.

More precisely, for instance, if secondary sense markers were used under a certain letter of a literal enumeration, and *after* the last letter

in the enumeration it is used a primary sense marker (thus higher with at least one unit in comparison to secondary markers), then the literal enumeration cycle at hand *can be closed* (one may not continue the literal enumeration refinement with the next letter in the alphabetic order).

We remind that we met a somehow similar (but more complex) problem for modeling the thesaurus-dictionary **DAR**, where the literal enumeration and the sense refinement introduced by the *NewPrg* (New Paragraph) typographic marker defining new senses (in various contexts) were calling each other [13], [14], [15]. The solution was there to introduce a special, *numeral enumeration* (with Roman small numerals, *LatSmaNumb_Enum*) for the sense markers *NewPrg*, then to close the finite mutual calls relying on the *enumeration closing condition* applied to several levels of sense description:

- the *literal enumeration* closing condition when this enumeration is developed *inside* a sense defined by a *single NewPrg* marker;
- the *numeral enumeration* closing condition when several *NewPrg* markers, encoded with the implicit, small Latin numbers *LatSmaNumb_Enum*, are developed within a sense described by a *single* small Latin letter of a literal enumeration *LatSmaLett_Enum*;
- once again the *literal enumeration* closing condition, when this enumeration is developed within a single, primary or secondary, regent sense.

Thus, we have here a *double enumeration*, a literal and a numeral one (the latter, generated by *NewPrg* markers), interleaving each other, each one with its enumeration closing condition. The entry **CAL** in **DAR** thesaurus-dictionary illustrates the exposed situation [15], [14], [28].

5 Atomic Sense Definitions and Example-To-Definitions in DMLRL: Their Dependency Hypergraph

5.1 DMLRL Specific Markers

(i). **The *tilda* “~” marker.** The role of this **DMLRL**-specific sense marker is to introduce a package of at least one definition of *TildaDef* type, with the aim of detailing the meaning of the sense definitions. The *TildaDef* package can be initiated at any level on the sense tree of a **DMLRL** entry, including the root-sense level of the word-lemma. *TildaDef*, *RefDef*, together with the *RegDef* most common device of sense description, provides the set of *autonomous definitions* in **DMLRL** (see the taxonomy in subsection 5.2). Subsection 3.3 describes in detail the role of **DMLRL**-specific “~” marker.

(ii). **The *traverse* “□” marker.** In **DMLRL**, this marker has several functions at the level of *atomic definitions* [11], [12], [14]:
 (1) The “*traverse*” sense marker is used to separate the author’s *example text* (called *DictExem* in **DMLRL**) from the *quoted text example* that follows (denoted *DefExem*, as in **DLR-DAR**), both (possibly) preceded by *specifying definitions* (*SpecDefs*, *SpSpecDefs*, or other ones).
 (2) The *traverse* marker “□” is also employed in **DMLRL** for displaying certain grammatical forms of the word-lemma. See also subsection 5.1, (E8: *DictExem*). Examples are [18 :780]:

БРОСОК, с к а, м. 1. Резкий взмах руки (рук), благодаря которому перемещается в воздухе что-л., находившееся в ней (в них). *Граната, разорвавшись при броске, оторвала мальчику правую кисть.* Коптяева, Дружба. □ **Б р о с к о м**, в знач. нареч. *Правой рукой он [рыбак] брал лежащую на парاپете полубесформенную массу осьминога и резким броском кидал ее на камни парапета.*

ВИОЛОНЧЕЛЬ, и ж. Смычковый четырехструнный инструмент, средний по регистру и размерам между скрипкой и контрабасом. *Партии альты и виолончели были в руках учителей*

музыкальной школы. Федин, Братья. □ Устар. Виолончель, я, м. Мы присутствуем при последних усилиях борьбы виолончеля за свое самостоятельное существование. Чайков. Третья неделя концертн. сезона.

(iii). The *asterisk “*”* marker. The task of this DMLRL-specific marker is to introduce a citation containing the use of the entry word-lemma with its *figurative meaning*. E.g. [18 :772-773]:

БРОСАТЬ, а ю, а е ш ь, несов.; **бросить**, б р о ш у, б р о с и ш ь, прич. страд, прош. б р о ш е н н ы й, а я, о е, сов.; *перех.* и *неперех.* 1. *Перех.* Резким движением, взмахом заставлять перемещаться в воздухе в каком-л. направлении копн, что-л.; кидать (в 1 знач.). *Бросить камень, палку.* [Чацкий:] *Кричали женичины: ура! И в воздух чепчики бросали!* Гриб. Горе от ума. Иногда аппетит [Прасковьи Павловны] даже совсем пропал, и она с досадой бросала на стол вилку и ножик. Салт. Сатиры в прозе. [Доктор] бросал мне стул, который я должна была поймать за ножки и бросить обратно. Кавер. Два капит. Войдя в избу, Михаил поставил на пол плетеную из бересты корзину, .. бросил к кровати мешок с валенками. Ф. Абрам. Две зимы и три лета. *Море глухо шумело, бросая на песчаную отмель гряды пенившихся волн. Мам.-Сиб, Вокруг раки, куста. Ветер бросал горсти листьев на стол, на койку, на прл. Паустов. Желт. цвет.

(iv). The *one-oblique-bar “/”* marker. This marker joins pairs of paradigmatic alternatives for the basic form of the entry word-lemma. E.g. [18 :773]:

... .. // В спортивной борьбе — вынуждать противника падать, касаясь лопатками ковра, земли. Борьба велась без приза, по просьбе дирекции, и Арбузов два раза бросал англичанина, почти шутя, редкими и эффектными трюками, которые он не рискнул бы употребить в состязании с мало-мальски опасным борцом. Купр. В цирке. ~ Бросать/бросить грязь, грязью в кого-л. См. Грязь. Бросать/бросить кого-, что-л. за борт. См. 1. Борт. *Жребий брошен.* См. Жребий. Бросать/бросить

камень, **камнем** в кого-л. См. Камень. **Бросать/бросить**
камешки в чей-л. **огород**. См. Камешек. **Бросать/бросить**
перчатку. См. Перчатка. **Бросать/бросить** что-л. **на чашу**
весов. См. Чаша.

2. *Перех.* Разводя руки, пальцы, выпускать, переставать
 держать что-л.

5.2 Atomic Definitions and Examples-to-Definitions in DMLRL: Taxonomies, Sequencing, and Dependencies

The definition types received specific functional roles in describing the meanings under **DLR** primary and secondary senses [11]. For the atomic senses / definitions, two taxonomies have been proposed in [12], [14], [15], to be used not only for **DLR-DAR** but also for **TLF**, **DWB**, **GWB**. Adapted and applied here to the **DMLRL** dictionary, the first taxonomy contains the following classes:

(**obli**) *obligatory definitions*, which are the *MorfDefs* and, for each **DMLRL** entry, *one* of the following three definitions, *RegDef*, *TildaDef*, (not exclusively when *RegDef* is present), or *RefDef*. The meaning of obligatory definitions is that there are no entries to have no *MorfDef*, and (at least) *one* of the *RegDef*, *TildaDef*, or *RefDef* definitions.

(**opti**) *optional definitions / examples-to-definition(s)* in **DMLRL**: *SpecDef*, *SpSpecDef*, *TildaDef* (when a *RegDef* is present), *RefDef*, *LexVarDef*, *DictExem*, and *DefExem*, whose presence is optional, as modifiers for an obligatory sense / definition.

A complementary taxonomy classifies **DMLRL** sense definitions and examples-to-definitions in:

(**auto**) *autonomous definitions*: *RegDef*, *TildaDef*, *RefDef*, and *LexVarDef*, meaning that these definitions have a stand-alone role in introducing **DMLRL** senses;

(**cont**) *contingent definitions / examples-to-definitions*: *MorfDef*, *SpecDef*, *SpSpecDef*, *LexVarDef*, *TildaDef* (when a *RegDef* is present), *DictExem* and *DefExem*, which do not have an independent, self-

determining meaning, but (possibly) playing the role of adjuncts, *i.e.* modifiers to some other definitions (including themselves).

MorfDef is obligatory at the root level of any **DLR** entry (*except* when the entry is defined by *RefDef*), being inherited (by default, when not present) on the lower levels of the entry sense tree. *MorfDef* is both an obligatory (at the root level) and also a *contingent* definition, when placed in front of an *autonomous* definition.

MorfDefs, *SpecDefs*, *SpSpecDefs*, *LexVarDefs*, *DictExems* and *DefExems* are *contingent* definitions since they cannot define a (sub)sense in an autonomous manner but they serve as auxiliary adjuncts to modify, to complete either autonomous definitions or other contingent definitions.

The lexicographic modeling of **DMLRL** for the parsing method of SCD *configurations* has to reveal at the beginning the entry segments (the *first* SCD configuration), the main segment of sense description being refined by primary and secondary senses, with their markers and dependencies, and their (possible) recursive relationship to literal enumeration (the *second* SCD configuration), whose image is the hypergraph in Fig. 1). The final level of lexical-semantics refinement is represented by the *third* SCD configuration, consisting of atomic sense definitions, examples to **DMLRL** autonomous definitions, their specific (sometimes, complex) markers, their sequencing and dependencies, their (autonomous / contingent and / or obligatory / optional) lexical-semantic role within a **DMLRL** entry. The third SCD configuration of **DMLRL** is illustrated in Fig. 2 below, a marker class dependency hypergraph, interconnected with that one in Fig. 1, and established for *the first time* at this level of specification for atomic sense definitions, among the studied large dictionaries **DLR**, **DAR**, **TLF**, **DWB**, **GWB** [15].

Trying to keep as close and unitary as possible to the already existing lexicographic SCD modeling of the atomic definitions and examples for **DLR-DAR**, **TLF**, **DWB-GWB**, we outline the following **DMLRL** atomic senses definitions, examples-to-definitions, their markers and dependencies [10], [11], [13], [15], [1], [16]. Each atomic sense definition is classified accordingly to the taxonomies proposed

above in this subsection (based on [12], [14], [15]). We found (until the current stage of **DMLRL** lexicographic investigation) that it is necessary to operate with the following **DMLRL** atomic sense definitions and examples-to-definitions:

(D1) *MorfDef* (*Morphologic Definition*); *Obligatory* and *Contingent* definition. When non-present, it should be inherited from a regent or a higher-level sense. It is written with Times New Roman, Italics font.

(D2) *SpecDef* (*Specification Definition*); *Contingent* and *Optional* definition. This is a *modifying* type definition applied in a cyclic or recursive manner to an autonomous definition. It is written with Times New Roman, Italic font. The expressions representing *SpecDefs* are usually abbreviated, reserved words.

(D3) *SpSpecDef* (*Spaced Specification Definition*); Similar to *SpecDef* but written with spaced-characters. *Internal reference* (inside the same entry), *external reference* (to another **DMLRL** entry), *morphological suffixes* or lexical variants are written, in certain contexts, with spaced-characters. See also *RefDefs*.

(D4) *RegDef* (*Regular Definition*); *Autonomous* and *Obligatory* definition. It is written with Times New Roman, Regular font. This is the basic tool to describe the semantic lexical-meaning of an entry sense / subsense in **DMLRL** (and in the largest majority of other dictionaries).

(D5) *TildaDef* (*Tilda-marker Definition*); *Autonomous* and *Optional* definition. Its description is enclosed in subsection 3.3.

(D6) *RefDef* (*Reference Definition*); *Autonomous* and *Optional* definition. *RefDefs* are *external references*, frequently met as constitutive part of the *TildaDef* definition package, or *internal references* to an entry sense (including the root-sense) inside which such a reference is used. We notice that all *RefDefs* are *SpSpecDefs* but the reverse is not true. See (D2) and (D3) below for typical examples.

(D7) *LexVarDef* (*Lexical-Variant Definition*); *Contingent* and *Optional* definition, used to provide lexical variation(s) to the entry-word. It is written with *bolded font*, and met within a *MorfDef*, when the meaning of the lexical variant is the same as that of the word-lemma.

(E8) *DictExem* (*Dictionary authors' Example*); *Contingent* and *Optional* example. This type of examples is given by the **DMLRL** dictionary authors to support the refinement of semantic explanations to autonomous definitions assigned to entry senses. *DictExems* usually follow an autonomous definition and are separated from *DefExems* by the traverse “□” **DMLRL** specific marker (see also subsection 5.1-(iii)).

(E9) *DefExem* (*Definition Example*); *Contingent* and *Optional* example. It is very similar to *DefExem* in **DLR-DAR** dictionaries [11], [15]. This type of examples represents quotations, text excerpts from bibliographic sources, with the role of refining and completing the meanings of autonomous definition(s) assigned to a (sub)sense of an entry. To each *DefExem* is associated a *sigle*, i.e. the *reference* of *DefExem* citation excerpt to its *bibliographic source(s)* or authorship.

The following further specifications and exemplifications concerning the above **DMLRL** atomic senses / definitions and their markers [1], [16] are considered. The relevant text of **DMLRL** definitions or examples-to-definitions at hand is highlighted in gray.

(D1: *MorfDef*) The *morphologic definitions* *MorfDefs* can form even a morpho-lexical package / segment in a **DMLRL** entry (see subsection 3.1) and describe the morphological categories at different levels of the entry sense tree. In general, the first element of a dictionary entry is a *MorfDef* that specifies certain syntactic categories, each one with its characteristic morphological-syntactic features. For **DMLRL**, the part-of-speech category of the word-lemma is *not present explicitly* but deduced from and described by its specific linguistic features; e.g., “м.” (masculine), “Перен.” (figurative) for nouns; “аю, аешь, несов.; **бросить**, брошу, бросишь, прич. страд, прош. брошенный, ая, ое, сов.; перех. и неперех.” contains flexional forms, reference to a sibling form (**бросить**, in bold and distinct font), transitive (*перех.*) and intransitive (*неперех.*) information for verbs etc. If *MorfDef* is missing at a (sub)sense level, then it is *inherited* implicitly from the regent or higher-level sense endowed with a *MorfDef* definition. Often, *MorfDef* is followed by *SpecDefs*, even when, by inheritance, it is missing overtly (as in the examples that follow).

АВАНС, а, м. **1.** Деньги, а также продукты, товары, выдаваемые а счет предстоящих платежей. *Получать аванс.,...* **2.** *Перен.* О том, что заранее дано или обещано и что необходимо оправдать, подтвердить в будущем.,... **3.** *Только мн. Перен. Устар.* О знаках внимания, поведении, вселяющих надежды на расположение, симпатию и т.п. ...

БРОСАТЬ, а ю, а е ш ь, *несов.;* **бросить**, б р о ш у, б р о с и ш ь, *прич. страд, прош. б р о ш е н н ы й, а я, о е, сов.; перех. и неперех.*

1. *Перех.* Резким движением, взмахом заставлять перемещаться в воздухе в каком-л. направлении копн, что-л.; кидать (в 1 знач.). *Бросить камень, палку.* [Чацкий:] *Кричали женщины: ура! И в воздух чепчики бросали!* Гриб. Горе от ума.

(D2: *SpecDef*) The *specification definitions SpecDefs* are various types of linguistic information (morphologic, syntactic, semantic, pragmatic, discursive, stylistic etc.) which refine and concentrate the meaning of the word, phrase, or text definition at hand. *SpecDefs* are written in Times New Roman, Italic font, many of them are *abbreviations* and *reserved* words (“*Снег.*”, “*Перен.*”, “*Разг.*” etc.), or parenthesized descriptions specifying different contexts of use within **DMLRL** senses. Functionally working as modifier expressions to be applied to the sense-subsense described, *SpecDefs* are both *contingent* and *optional* definitions. *SpecDefs* are present at any level of the entry sense tree. They are frequently enclosed in and associated with *MorfDefs*. Examples of *SpecDefs* (and also *SpSpecDefs*):

... ..

11. *Перех. и неперех. Разг.* Достигать.цели, добиваться успеха посредством чего-л. О нем [Прокофии] *стали говорить тогда:— Новый-то круто берет, а!., новый-то что удумал.*

4. *Только Зл. Разг.* Ловиться на удочку (о рыбе). *Ходил рыбачить на озеро. Плотва хорошо бралась, только успевай червя насаживать.* Горыш. Тридц. лет спустя.

БРОСАТЬ, а ю, а е ш ь, *несов.;* **бросить**, б р о ш у, б р о с и ш ь, *прич. страд, прош. б р о ш е н н ы й, а я, о е, сов.; перех. и неперех.*

1. **Перех.** Резким движением, взмахом заставлять перемещаться в воздухе в каком-л. направлении копн, что-л.; кидать (в 1 знач.). *Бросить камень, палку.* □ [Чацкий:] *Кричали женщины: ура! И в воздух чепчики бросали!*

БРАТЬ, беру, берешь, **прош.** брал, ла, ло, **несов., перех. и неперех.**, (**сов.** в з я т ь). 1. Захватывать рукой, руками; принимать в руки. *Брать ложку. Брать со стола книгу. Откинув локонь от милого чела, Сама из рук моих свирель она брала.* Пушкин. Муза. [Доктор] *брал его руку, отсчитывал пульс.* Горбачев. Мое поколение. ◇ **Б р а т ь** чем-л. [Сахар] *приходилось брать щипчиками.* В. Катаев, Хуторок в степи. ◇ **Б р а т ь** руками, в руки что-л. *Он тянулся за дудкой, брал ее дрожащими руками и прикладывал к губам.* Король. Слеп, музыкант.

БРАТСТВО, а, **ср.** 1. Содружество, единение, союз, основанные на общности целей, взглядов, принципов и т. п. [Пьер] *твердо верил в возможность братства людей, соединенных с целью поддерживать друг друга на пути добродетели.* Л. Толст. Война и мир. // **Собир.** Люди, объединенные общей целью, общим делом ит. п. ◇ **Б р а т с т в о** какое-л., кого-л. *Студенческое братство,* □ *Газетное братство распадалось на целый ряд категорий: передовики, фельетонисты, хроникеры, заведующие отделами вообще.* Мам.-Сиб. Черты из жизни Пепко.

(D3: **SpSpecDef**) The *spaced-specification definitions* (*SpSpecDefs*) are used, in general, either to specify morphological / lexical forms and variants, or to *internally* (inside the same entry) and *externally* (to another **DMLRL** entry or entry sense) refer a **DMLRL** entry sense / subsense. *SpSpecDefs* may occur not only in the sense description segment but also into the morphological, *TildaDef*, and etymological description segments. It is important to mention that a *SpSpecDef* expression in **DMLRL** is rather distinct from that defined in **DLR-DAR** [11], [14], [15]. *SpSpecDefs* in **DMLRL** are employed to describe the following situations [1]:

(i) Collocations of the word-lemma in various expressions. *E.g.:*

АБРИКОСОВЫЙ, а я, о е. 1. Относящийся к абрикосу, абрикосам (в 1 знач.). ◇ **А б р и к о с о в о е** дерево. То же, что абрикос.

(ii) Lexical and syntactic-phrase variants of the entry-word, as in:

АВАНС, а, м. 1. Деньги, а также продукты, товары, выдаваемые а счѣт предстоящих платежей. *Получать аванс...* □ **А в а н с о м**, в знач. нареч. Вперѣд, заранее. [На дачу] *пошли деньги взятые авансом у издателя*. В. Андреева, Дом на Чѣрн. Речке.

(iii) *Internal references* (inside the same entry) and *external references* (to another **DMLRL** entry), thus *RefDefs*, are also *SpSpecDefs* in **DMLRL**, i.e. Times New Roman, regular, spaced-character written. The example that follows contains (grayed) external *RefDefs* (ending the previous entry that precedes “**БРАТЬСЯ**”), morphological derivations, and internal *RefDefs* (inside the “**БРАТЬСЯ**” entry).

... См. 2. М у ш к а **Братъ** кого-, что-л. *на прицел*. См. **П р и ц е л**. **Братъ** кого-л. *на пушку*. См. 1. П у ш к а. **Братъ** кого-, что-л. *под обстрел*. См. **О б с т р е л**.

— Срезневский: б р а т и; Поликарпов, 1704: б е р у; Вейсманн, 1731, с 154: б р а т и денги; Росс Целлариус 1771, с 9: б е р у, б р а т ь.

БРАТЬСЯ, б е р у с ь, б е р е ш ь с я, *прош.* б р а л с я, б р а л а с ь, б р а л о с ь и б р а л о с ь *несов.* (сов. **в з я т ь с я**). 1. Захватывать что-л., хвататься за что-л., касаться чего-л. Руками. ◇ **Б р а т ь с я** за что-л. *Братся за поручни. Братся за голову, за подбородок.* □ *Он умолкал, иногда надолго. Справляясь с волнением, он крепко брался за спинку стула.* Кавер. Откр. книга. ◇ **Б р а т ь с я** руками, пальцами

(iv) Flexion suffixes of **DMLRL** entry-word, usually met in the morphological segment. *E.g.*:

ВИРТУОЗНЫЙ, а я, о е; з е н, з н а, з н о. Относящ. к виртуозу, свойственный ему. *Виртуозное исполнение.*

(v) The use of the spaced-character entry-word (lemma or derivations) within specific phrases, as in:

БЫЧИЙ, *ь е, ь я*. 1. ... 2. ◇ **БЫЧИЙ** глаз. *Разг.* Болезненное растяжение и выпячивание глазного яблока. ◇ **БЫЧЬЕ** сердце. *Разг.* Болезненно увеличенное (в размерах и по массе) сердце.

(D4: *RegDef*) *RegDef* is an *autonomous* and *obligatory* definition, written with Times New Roman, Regular font. *RegDef* is the standard device to describe the semantic lexical-meaning of an entry sense / subsense in **DMLRL** (and in many other thesaurus-dictionaries, including **DLR-DAR**). Sample of (grayed) *RegDef*:

БЫТОПИСАНИЕ, *я, ср.* 1. *Устар.* **Историческое описание, история.** Он рыться не имел охоты В хронологической пыли Бытописания земли. Пушкин. ...

(D5: *TildaDef*) The *TildaDef* package / segment of definitions is described in subsection 3.3.

(D6: *RefDef*) *RefDefs* are autonomous, *external references*, frequently met as constitutive parts of the *TildaDef* definition (or package), or *internal references* to an entry sense (including the root-sense) inside which such a reference is used. *RefDefs* are written with Times New Roman, regular, spaced-characters, thus they all are *SpSpecDefs*; the reverse is not true. (D2) and (D3) contain instances of *RefDefs*. In the example (D3: *SpSpecDef*)-(iii) given above, the first four grey fields are external *RefDefs* and the last three ones are internal *RefDefs*. The autonomous role of external *RefDefs* is shown in the following examples [18 :771]:

БРОНХ. См. **Б р ó н х и**.

... ..

БРОНХИО́ЛА. См. **Б р о н х и б л ы**.

(D7: *LexVarDef*) The *LexVarDef* (*Lexical-Variant Definition*) is a *contingent* and *optional* definition, used to provide lexical variation(s) to the entry word-lemma. It is written with small, regular characters, *bolded font*, and met within a *MorfDef* (when the meaning of the lexical variant is the same as that of the word-lemma; *e.g.* the pairs in the fol-

lowing examples: **ВИЛЯТЬ-вильнуть**, **БРОСАТЬ-бросить** [18:860, 772].

ВИЛЯТЬ, я ю, я е ш ь, *несов.*; **вильнуть**,
БРОСАТЬ, а ю, а е ш ь, *несов.*; **бросить**, б р о ш у, б р о с и ш ь,
прич. страд, прош. б р о ш е н н ы й, а я, о е, *сов.*; *перех. и неперех.*
1. Перех. Резким движением, взмахом заставлять перемещаться в
воздухе

(E8: *DictExem*) *DictExems* are examples given by **DMLRL** dictionary to support the refinement of semantic explanations to autonomous definitions assigned to the entry senses. *DictExems* usually follow an autonomous definition and are separated from the other *DefExems* by the *traverse* “□” **DMLRL** specific marker (subsection 5.1-(ii) describes the *traverse* marker role). The difference between a *DictExem* and a *DefExem* is that the former do not bear a *sigle*, *i.e.* the reference to the bibliographic source of the example-to-definitions (this one is just the dictionary authorship), while *DefExem* has to provide its bibliographic source, *viz.* its *sigle(s)*. When both *DictExems* and *DefExems* are present, the former are located always as the first ones, followed by the *traverse* “□” marker, which signals the end of *DictExems* sequence and the beginning of the *DefExems* block. Numerous samples of *DictExems* and *DefExems* are already shown in the paper.

(E9: *DefExem*) The role and structure of a *DefExem* (*Definition Example*) is to support and refine a lexical-semantics sense definition, already outlined in (E8: *DictExem*) above. *DefExem* in **DMLRL** is actually the same *DefExem* example-to-definition that is working for **DLR-DAR** dictionaries [11], [15].

6 Conclusions

The special *features* of the *new parsing method* with **SCD configurations** (SCD-*configs*) are: • The SCD-*configs* method for dictionary entry parsing is based on sense marker classes, their lexical-semantics dependency (*i.e.* sense structure subsumption), and *procedural hypergraphs* reflecting the sense marker class sequencing and dependencies

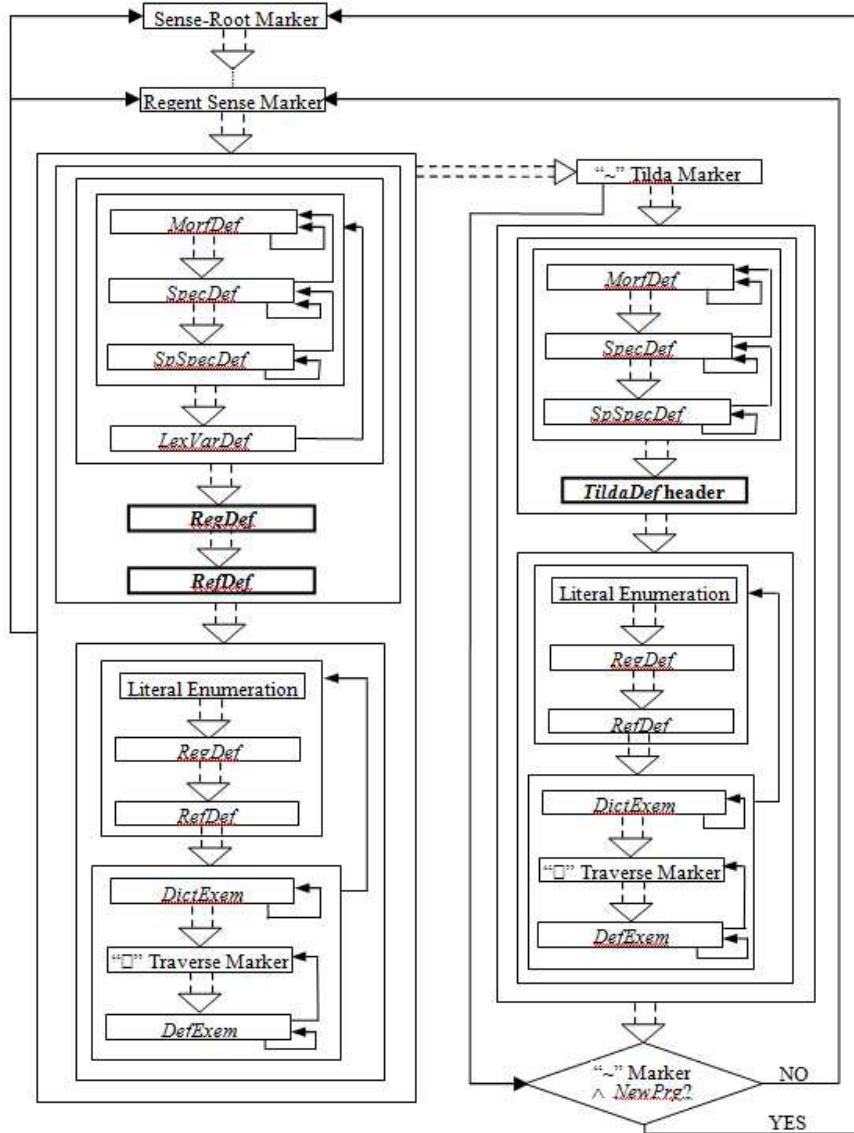


Figure 2. *RegDef* block and *TildaDef* block sequences and dependencies for **DMLRL** atomic sense definitions and examples-to-definitions

for each SCD configuration [11], [15]. • SCD-*configs* is a completely *formal grammar-free* approach which involves simple, efficient (weeks-time adaptable), thus portable modeling and programs [15]. • The method of SCD-*configs* for dictionary entry parsing is derived from the more general SCD linguistic theory and parsing strategy for natural language free text [7], [5], [3], [4]. • The main drawback of the currently existing parsing methods for dictionary entry parsing is that the sense tree construction of each entry is recursively embedded and mixed within the definition parsing procedures [6]. • To overcome this essential problem, the SCD-*configs* separate and run sequentially, on independent levels (*viz.* SCD configurations), the processes of *lexicographic segment recognition*, *sense tree extraction* (for entry senses defined by explicit marker classes), and *atomic definition parsing*. • This makes the whole *dictionary entry parsing* process with SCD-*configs* to be *optimal* [15], [11].

The main results of this paper consist in identification and behavior description of the three SCD configurations that are specific to **DMLRL** dictionary: SCD-*config1* shows the linear sequence of **DMLRL** lexicographic segments, while SCD-*config2* deals with sense marker classes associated to the primary and secondary senses in **DMLRL** and to their dependencies, displayed as the hypergraph in Fig. 1. Already pointed out in subsection 4.3, the solution to the problem of recursive calls between the secondary senses ($//$ and \diamond) and the refinement procedure of literal enumeration is the *enumeration closing condition*. The SCD-*config3* is represented in Fig. 2 as the hypergraph of the atomic sense / definition markers in **DMLRL** and *interconnected* with the hypergraph in Fig. 1. That one gives the dependency relationships among the higher-order sense marker classes, handing down from the root-sense, through primary and secondary senses, continued with the dependency hypergraph for the lower and atomic senses / definitions, represented in Fig. 2. When structurally accomplished, **DMLRL** lower-level senses are raising up, called by higher-level sense markers, until the structure of the entry sense tree is completed.

We provide in this paper the atomic definitions and examples-to-definitions that contribute to sense construction, their obligatory, au-

tonomous, contingent and / or optional functional role, described with their marker class sequences and dependencies. The type of dependency hypergraph in Fig. 2 is displayed for the *first time*, at this level of lexical-semantics specification, among the other similar dictionaries investigated for lexicographic modeling and parsing [15]. The **DMLRL** lexicographic segments, along with the higher-level marker class dependencies and hypergraph behavior in Fig. 1, procedurally interconnected with the hypergraph in Fig. 2, represent the complete lexicographic modeling of the three SCD configurations, which can ensure a high-performance parsing process of **DMLRL** dictionary, as proved for similar or more complex thesaurus-dictionaries [14], [15].

References

- [1] Burcă, Eugenia (2011): *Parsing the Dictionary of Modern Literary Russian Language (DMLRL) using the Method of Segmentation-Cohesion-Dependency Configurations*, Institute of Mathematics and Computer Science, Chişinău, Rep. of Moldova, 12 p. (in Romanian, draft paper).
- [2] Cristea, D., Răschip, M., Forăscu, C., Haja, G., Florescu, C., Aldea, B., Dănilă, E. (2007): *The Digital Form of the Thesaurus Dictionary of the Romanian Language*. In Proceedings of the 4th International IEEE Conference SpeD 2007.
- [3] Curteanu, Neculai (1994): *From Morphology to Discourse Through Marker Structures in the SCD Parsing Strategy. A Marker Hierarchy-Based Approach*. Language and Cybernetics, INTERK-IBERNETIK'93, Akademia Libroservo, Prague, Czech Republic, pp. 61–73.
- [4] Curteanu, Neculai, G. Holban (1996): *The SCD Linguistic Strategy Applied to the Analysis and Generation of Romanian*. In the volume "Language and Technology", (Ed. Dan Tufiş), The Romanian Academy Editorial House, Bucharest, pp. 169-176 (In Romanian).

- [5] Curteanu, N., D. Gălea, C. Butnariu, C. Bolea (2004): *Marcu's Clause-like Discourse Segmentation Algorithm and SCD Clause Segmentation-based Parsing*, Proceedings ECIT-2004 Conference, pp. 59–86, Iași, România.
- [6] Curteanu, N., E. Amihăesei (2004): *Grammar-based Java Parsers for DEX and DTLR Romanian Dictionaries*. ECIT-2004 Conference, Iasi, Romania.
- [7] Curteanu, N. (2006): *Local and Global Parsing with Functional FXbar Theory and SCD Linguistic Strategy*. (I.+II.), Computer Science Journal of Moldova, Academy of Science of Moldova, Vol. 14 no. 1 (40):74–102 and no. 2 (41):155–182.
- [8] Curteanu, N., D. Trandabăț, G. Pavel, C. Vereștiuc, C. Bolea (2007): *eDTLR Project – The Romanian Thesaurus-Dictionary in Electronic Format*. Research Report to the PNCDI II Project No. 91_013/18.09.2007, Stage on 2007 (in Romanian).
- [9] Curteanu, N., G. Pavel, C. Vereștiuc, D. Trandabăț (2008): *eDTLR Parsing with Lexicographic Grammars in the JavaCC Framework. The Current Stage, Problems, and Development Solutions*. In Proceedings of the Workshop on Linguistic Resources and Instrument for Romanian Language Processing – ConsILR-2007, (Ed. I. Pistol, D. Cristea, D. Tufiş), The "Al.I. Cuza" University Editorial House, Iași, ISSN: 1843-911X, pp. 87–96 (in Romanian).
- [10] Curteanu, N., D. Trandabăț, A. Moruz, C. Bolea, M. Husarciuc (2008): *Parsing the Romanian Language Thesaurus Dictionary (new format) at Sense Trees and Definitions, with the Method of SCD Configurations*. Research Report to the Grant Project PNCDI 2, Nr. 91_013/18.09.2007, Stage on 2008 (in Romanian).
- [11] Curteanu, N., Moruz, A., Trandabăț, D. (2008): *Extracting Sense Trees from the Romanian Thesaurus by Sense Segmentation & Dependency Parsing*, Proceedings of CogAlex-I Workshop, COLING 2008, Manchester, United Kingdom, pp. 55–63, ISBN 978-1-905593-56-9.

- [12] Curteanu, N., A. Moruz, D. Trandabăţ, Cecilia Bolea, Mădălina Spătaru, Maria Husarciuc (2009). *Sense Tree Parsing and Definition Segmentation in the eDTLR Thesaurus-Dictionary eDTLR*, In Proceedings of the Workshop on Linguistic Resources and Instrument for Romanian Language Processing – ConsILR-2008, (Ed. D. Trandabăţ, D. Cristea, D. Tufiş), Editura Univ. “Al.I. Cuza” Iaşi, ISSN: 1843-911X, pp. 65–74 (in Romanian).
- [13] Curteanu, N., A. Moruz, D. Trandabăţ, C. Bolea (2009): *Parsing the Romanian Academy Thesaurus Dictionary (old format) and Romanian Language Thesaurus Dictionary (new format) at Sense Trees and Definitions, with the Method of SCD Configurations*. Research Report to the Grant Project PNCDI 2, Nr. 91_013/18.09.2007, Stage on 2009 (in Romanian).
- [14] Curteanu, N., A. Moruz, D. Trandabăţ (2010): *Comparative Parsing of the Romanian, French, and German Thesaurus-Dictionaries*. In Proceedings of the Workshop on Linguistic Resources and Instrument for Romanian Language Processing, (Ed. A. Iftene, H.N. Teodorescu, D. Cristea, D. Tufiş), Editura Univ. “Al.I. Cuza” Iaşi, ISSN: 1843-911X, pp. 113–122 (in Romanian).
- [15] Curteanu, N., Trandabăţ, D., Moruz, A. (2010): *An Optimal and Portable Parsing Method for Romanian, French, and German Large Dictionaries*, Proceedings of COGALEX-II Workshop, COLING-2010, Beijing, China, August 2010, pp. 38–47.
- [16] Curteanu, Neculai (2011): *The SCD-based Lexicographic Modeling of DMLRL – Dictionary of Modern Literary Russian Language*, Research Report, Institute of Computer Science, Romanian Academy, Iasi Branch, June 2011 (in Romanian).
- [17] Das Woerterbuch-Netz (2010): <http://germazope.uni-trier.de/Projects/WBB/woerterbuecher/>
- [18] Dictionary of Modern Literary Russian Language (20 volumes – 1994): Editorial House: M.: Russkii Iazyk; Second edition, revised

- and supplemented, 864 p; 1991 – 1994. ISBN: 5-200-01068-3 (in Russian).
- [19] DLR revision committee. (1952). *Coding rules for DLR* (in Romanian). Romanian Academy, Institute of Philology, Bucharest.
- [20] Erjavec, T, Evans, R., Ide, N., Kilgariff A., (2000): The CONCEDE Model for Lexical Databases. Research Report on TEI-CONCEDE LDB Project, Univ. of Ljubljana, Slovenia.
- [21] Hauser, R., Storrer, A. (1993): *Dictionary Entry Parsing Using the LexParse System*. Lexikographica 9 (1993), 174–219.
- [22] Kammerer, M. (2000): *Wörterbuchparsing Grundsätzliche Überlegungen und ein Kurzbericht über praktische Erfahrungen*, <http://www.matthias-kammerer.de/content/WBParsing.pdf>
- [23] Le Trésor de la Langue Française informatisé (2010). <http://atilf.atilf.fr/tlf.htm>
- [24] Lemnitzer, L., Kunze, C. (2005): *Dictionary Entry Parsing*, ESS-LLI 2005.
- [25] Marcu, Daniel. 1997: *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*, Ph.D. Thesis, Univ. of Toronto, Canada, pp. 331.
- [26] Neff, M., Boguraev, B. (1989) *Dictionaries, Dictionary Grammars and Dictionary Entry Parsing*, Proc. of the 27th annual meeting on Association for Computational Linguistics Vancouver, British Columbia, Canada Pages: 91 – 101.
- [27] ORDA License Copyright Registration (2011). Owner: Curteanu, Neculai. Title: *The SCD (Segmentation-Cohesion-Dependency) Lexicographic Modeling and Parsing Strategy for Natural Language Text of Some Romanian, French, German, and Russian Thesaurus Dictionaries*, ORDA – The Romanian Copyright Office, RNO Registration No. 9134 / 25.07.2011.

- [28] Pușcariu, Sextil *et al.* (1906): Dictionary of the Romanian Language (Dictionary of the Romanian Academy – **DAR**), Bucharest, Edition 1940 (old format).
- [29] Tiktin, H. (1989): *Rumänisch-deutsches Wörterbuch*, 2., überarbeitete und ergänzte Auflage von Paul Miron. [Band I–III]. Otto Harrassowitz. Wiesbaden. I: 1986; II: 1988; III: 1989.
- [30] Tiktin, H. (2005): *Rumänisch-deutsches Wörterbuch*, 3., überarbeitete und ergänzte Auflage von Paul Miron und Elsa Lüder. Band I–III. Cluj-Napoca, Clusium. I: 2000; II: 2003; III: 2005.
- [31] Tufiş, D., Rotaru, G., Barbu, A.M. (1999): *Data Sampling, Lemma Selection and a Core Explanatory Dictionary of Romanian*. Proc. of the 5th International Workshop on Computational Lexicography COMPLEX, Pecs, Hungary, pp. 219–228, 1999.
- [32] Tufiş, Dan (2001): From Machine Readable Dictionaries to Lexical Databases, RACAI, Romanian Academy, Bucharest, Romania.
- [33] XCES TEI Standard, Variant P5 (2007): <http://www.tei-c.org/Guidelines/P5/>

Neculai Curteanu, Svetlana Cojocaru, Eugenia Burcă, Received March 30, 2012

Neculai Curteanu,
Institute of Computer Science, Romanian Academy, Iaşi Branch,
Str. Gh. Asachi, Nr. 3,
700483 Iaşi, România
E-mail: ncurteanu@yahoo.com

Svetlana Cojocaru, Eugenia Burcă,
Institute of Mathematics and Computer Science, Academy of Sciences of Moldova,
Str. Academiei nr. 5, Chişinău,
MD 2028, R. Moldova
E-mails: Svetlana.Cojocaru@math.md, eugenia_burca@yahoo.com

Using Test Case Mutation to Evaluate the Model of the User Interface

Izzat Alsmadi

Abstract

Mutation based testing is used to discover new possible errors in software applications. This is since in this testing approach, intentional incorrect lines of codes are injected to check the software ability to produce results that are different from the correct or original code. In this paper an automatic technique to generate valid and mutant test cases is proposed and developed. In most mutation techniques, one or more values or parameters in the specification, code, model, etc are intentionally modified and then test cases are generated to see if injected modifications can be detected. However, in this paper, test cases are mutated (i.e. mutants are generated from the test cases) after they are generated from the GUI model. Mutations are then applied to the GUI model to test its ability to kill those mutants by rejecting them. Typical to mutation testing, the goal of this approach is to discover possible errors or problems in the program that may not be discovered by other methods. A robust model is expected to differentiate between a valid and an invalid sequence of events. An automatic execution and verification technique is also developed to evaluate the test cases that were rejected by the model and calculate coverage based on the number of rejected test cases to the total number of test cases. Results showed that in user interfaces, and based on the nature of the mutation process implementation, mutation can find new areas or types of errors that may not be found using other approaches of testing.

Keywords: Mutation testing, Test case generation, test case execution and verification. Random test case generation, and GUI modelling.

1 Introduction

It is widely acknowledged that testing activities consume a significant amount of software project resources. This is why research projects in software testing focus on aspects that can reduce those expenses while maintain or improve coverage. Test automation techniques are used to achieve this goal. In order to use test automation, Artificial Intelligent (AI) algorithms are used to replace or simulate tester activities. Those activities include: test case generation, execution and verification. Mutation is a surplus testing activity used in general to improve test case generation and verification effectiveness. This is accomplished by changing a small part of the code or the specification. Test cases are then applied to see the test cases that can kill (i.e. discover) those mutants. In this paper, mutation is used to evaluate the reliability of the GUI model. In traditional code mutation processes, mutation coverage can show whether test cases would expose the use of wrong operators and also wrong operands. It works by reporting coverage of conditions derived by mutating (i.e. substituting) the program's expressions with alternate operators, such as "less than" substituted for "more than". In the traditional mutation, mutation is occurred to the code or the specification and test cases are expected to discover this mutation. In this paper, the process is reversed. Mutation occurs in test cases and the GUI model is expected to discover those mutations. Such approach may fall under model based testing techniques where the GUI model is tested for its ability to kill (i.e. reject, in the scope of this paper) wrong test cases.

Why would someone apply mutation to test case generation?! In GUI testing, most GUI components have one main event interaction. For example, a button main even interaction is the "double click", the textbox main interaction is "entering a text", the option item main interaction is selecting one or options, etc. A GUI abstraction model that considers the GUI components, their attributes and association with each other along with one main event for each component is developed [9, 10]. As such, the abstraction model considers both GUI structure and event models. In this model, test cases can be generated directly

```
<GUI-Forms><Root>GUI-Forms</Root>
<Parent-Form>frmDataDisplay</Parent-Form>
<Name>Settings</Name>
<Parent-Form>frmDataDisplay</Parent-Form>
<Name>Program</Name>
<GroupBox>
<Parent-Form>frmDataDisplay</Parent-Form>
<Name>GroupBox2</Name> </GroupBox>
<ListBox><Parent-Form>GroupBox2</Parent-Form>
<Name>lstViews</Name> </ListBox>
```

Figure 1. A simple GUI structure sample generated from an application in an XML format.

from the GUI structure file. Figure 1 shows a simple screen shot sample generated from an application for a GUI structure. The XML file (generated automatically from the application at run time using reflection; a reverse engineering process to discover the program GUI components from its executable) contains all GUI components along with each component parent. Test cases are then automatically generated from this model through traversing through GUI paths starting from the entry point to an end or leaf point. For each component, test cases are considering the component default event (in order to generate the test case that will be executed automatically).

Is it significantly useful to make an effort to inject errors in test cases and then make extra effort trying to find them ?! A mutation in a test case means that we will try to execute a test case with invalid sequence or combination of GUI components. Such sequence should not be executed successfully. This is somewhat similar to the specification based testing that tests the application using valid and invalid inputs. The application is expected to accept valid inputs and produce consistent results while rejecting invalid inputs and halt the execution. The next section introduces the related work. Section 3 lists the goals of this research and describes the work done toward those goals. Section

4 presents the conclusion and future work.

2 Related Work

In this literature survey, several relevant papers are discussed. Those papers discussed using mutation for evaluating coverage and test case effectiveness. All papers listed as references in this paper focus on generating mutation operators based on one aspect of software products and then evaluate test coverage or effectiveness from this mutation process. The major difference that distinguish one paper from the others is the software aspect that mutation operators are generated from (e.g. requirements, software model, code, state diagram, etc).

One of the prominent researchers in the area of testing in general and mutation testing in particular is Jeff Offutt at George Mason University (cs.gmu.edu/~offutt). He has several books, book chapters and relevant papers sole or with friends and students. He also developed the widely used mutation tool muJava (<http://cs.gmu.edu/~offutt/mujava/>). Examples of some of those contributions include: [1,2,3,4,5,6,7,8]. These papers discussed developing and using mutation tools such as Java and Mothra. They also discussed Mutation operators and using mutation in source code, Web applications and object oriented code. Coverage (e.g. code, and path) was a criterion to evaluate the effectiveness of mutation against it. Mutation can be divided based on the software stage where it occurs, or based on the software product component(s) upon which the mutation process occurs. For example, there are several papers that discuss: source code mutation (e.g. Java, object oriented code), Windows or Web mutation, test cases' mutation, database, integration testing, design or requirement mutation. The selection of the papers in this literature review of related work is based on selected variations of these different mutations.

In [1], Choi et al presented one of the earliest mutation based testing environment. Other examples of mutation tools include: Java, Clipse, Javalanche, Jumble, Certitude, Jester, Proteum, and SQLMutation.

In our paper, a new mutation tool is developed for the particular mutation in user interface components. The Mothra testing project was

initiated in 1986 by members of the Georgia Institute of Technology's software engineering research center. Mothra is written in FORTRAN and consists of a collection of individual tools, each of which implements a separate, independent function for the testing system. Examples of some of its mutation operators include relation operators' replacement. In original Mothra, the tool converts the tested code to an intermediate code in order to execute its mutated version by an interpreter.

In a recent paper [2], Mateo et al proposed mutation operators that are somewhat related to GUI components as some of these operators were trying to evaluate whether a component is interchanged with an earlier version of the same component which is something that may occur frequently especially with an evolving application. This system level mutation is a continuation for a work done in this area previously by Delamaro et al [14, 16, 17, and 24] work on MuJava. On the GUI level, the mutation operators proposed consider a small subset of the possible GUI mutants such as GUI components' position or order interchange, component deletion and modification. However, as the paper had a large scope, extensive evaluation and implementation of these mutations were not mentioned. As our paper came as an extension of a GUI test automation tool [9, and 10] which includes the automation of all GUI testing activities: generation, execution and verification, the tool is utilized and extended to generate GUI mutants along with implementing the ability to automatically execute these mutants and evaluate their results.

In [3], Offutt et al described how to use the information of equivalent mutation for the problem of some paths' feasibility. An equivalent mutant is the one that will always produce the same output as the original program, so no test case can kill it. This affects the mutation score and causes it always to be less than complete coverage. Earlier, in his dissertation, Offutt proposed using Constraint-Based Testing (CBT) for detecting equivalent mutants [5]. The paper presented a method to detect equivalent mutants in code through using constraints. These constraints are applied on the input domain to narrow its scope. GUI mutation has some possible equivalent mutation operators that will be

discussed in this paper.

Li et al evaluated mutation as a coverage criterion in comparison with other testing coverage criteria such as path, edge, etc. [6]. Coverage is a test case metric that is used to measure the ability of a particular test approach to cover one or more aspects of the software code that may include: statement, branch, path, etc. coverage. The study focus was on unit testing and showed that mutation can actually be more effective in terms of coverage from many other test criteria. Unlike mutation score, mutation coverage calculates the number of faults that can be detected using the mutation process. In our GUI testing approach, the GUI is serialized from the actual application dynamically using a reverse engineering process. (i.e. .NET reflection). As a result, the GUI model is represented by an XML file which includes the GUI components, hierarchy and attributes. Mutations are created based on the XML file and applied on the actual GUI during the execution process. However, since a GUI test case looks like a GUI path (e.g. File,Save,Exit), mutation can be also applied on or generated from the test cases. In typical cases, the program is mutated and the test cases are used to detect this mutation. However, in GUI, it is possible to reverse the process through generating a mutated test case (that may include for example an invalid GUI component) and then execute it on the application where its failure is an indication that the mutant is killed. If the mutant test case failed then this is a possibility of two: either the mutant test case is equivalent and looks like a normal test case to the GUI, or the test case is killable but the current state of the GUI failed to kill this mutant.

Offutt et al discussed a selective mutation process to reduce the expenses of mutation as the number of possible mutation operators for even a small program can be significant [7]. A selective mutant can be selected from many mutants if they produce the same results. Lee et al. presented a Web based scenario level mutation based on interaction scenarios written in an Interaction Specification Model (ISM) [8]. ISM is an XML based interaction constraint language. Message mutation can be applied in a wide range of applications that use messaging such as distributed systems, networks, etc.

Hierons paper represents another paper in specification or modeling mutation [24]. The paper discussed Finite State Machines (FSMs) mutation based on the basic elements of FSMs which are: states, events and transitions. In FSM, each state is recognized by preconditions which represent the constraints that are required to occur or be true in order for the transition to the target state to occur, and post conditions that represent the expected results from the state transition. In relation to this subject, we are planning to extend GUI mutation in future to cover GUI state based mutation aspects.

Other papers such as [11] and [14] discussed using mutation for evaluating states coverage. Examples of mutation operators for finite state machines include: event, arc, or output missing, extra or exchanged. For user interface mutation this also can be considered for GUI events interaction, however, the focus of our mutation operators here is on the GUI structural mutation.

There are some other papers such as [13 and 18] that used the mutation process as a technique for validating a particular testing algorithm or approach. Bradbury et al [13] used a subset of the Concurrency Mutation Analysis (ConMan) operators that are discussed in the authors' earlier paper [23]. However, by larger, model checking is a testing area where mutation is not thoroughly investigated. In this paper and in mutation research papers in general, one weakness of such studies is that same authors are the one who usually create mutations and create tests to detect them. If the goal of mutation is to try to imitate real life bugs and test abilities to detect them, the creation of mutation and the detection process should both be independent from each other.

Papers from [15, 16, 17, 18, 19, 20, 21 and 22] include examples of using mutation in other software artifacts such as: Web, object orientation and aspect oriented programming.

3 Goals and Approaches

3.1 Using Test Case Mutation to Evaluate the Model of the User Interface

Mutation is used in testing for various purposes. It is first used to inject faults into the system and measure the system or the test cases' ability to catch those mutants. As a result, mutants can indicate the effective test cases through their abilities to detect errors which allow us to eliminate ineffective test cases and improve test effectiveness. Mutation can be also used to indirectly generate test cases. Mutation testing can be also used to indirectly verify requirements. In mutation, three steps are accomplished: First mutants are created according to mutation operators; second program is executed with normal and mutant inputs. Finally verification and coverage analysis is implemented based on the percentage of mutants discovered. In this paper we will present mutation for the test cases. In our GUI model, each test case contains a sequence of GUI components. This means that the program executing those test cases will interact with those components consecutively with the right or typical type of interaction (e.g. a button click, a textbox type text, an option list option select, etc). Default types of interactions are defined for each control type. As such, the mutations that will be evaluated first are changing one component or widget in each test case. From previous knowledge, we know that if the mutated control is changed to a control in the same level, the new test cases can still be executed (however, it should produce a different behaviors). As such, we can divide the expected behaviors from test cases' mutation into 3 levels:

- It is expected that the majority of mutated test cases should be rejected as they will produce invalid test cases that will not fully and successfully be executed. Execution and Verification tool should be able to distinguish that such mutant test cases produced different results relative to the original ones.
- Some mutations will pass the validation process and produce a valid test case. However, they will produce different results or

behavior relative to the original test cases.

- It is expected that few mutations will not be killed at all as they will be valid and produce identical results compared to the original test case or the different behavior can't be distinguished or observed.

3.2 The Execution and Verification Process

Implementing and verification process for testing is one of the challenging processes that have several obstacles. An algorithm is developed to read test cases one by one and execute them on the actual application. Each GUI component is then tested to see if it is successfully executed or not. Once all test case components are successfully executed, the test case suite passes the execution and verification process. In such processes many problems can occur. Timing is one of the problems where one form or web page needs to wait for another execution to finish while it is expecting it earlier. Synchronization and dealing with multithreads are also other examples of such problems. In order to focus on evaluating and comparing original test cases with mutants, any test case from the original test fails the execution and the verification process will be eliminated. The importance of evaluating this execution and verification effectiveness is that it can test our mutation process by measuring the ratio of killed (i.e. invalid) mutants to valid ones. Any invalid component should not be executed successfully. Figure 2 shows a sample log of the execution and verification process output. The test list count shows the input GUI components to the execution process and the test execution count shows the number of GUI components that were successfully executed. The difference between the two numbers (i.e. test list count – test execution count) represents the number of GUI components that fails in the execution process.


```
test list count== FRMDATADISPLAY FRMCONNECT FRMDATADISPLAY
FRMCONNECT TABCONTROL1 TABSQLSERVER GBXSQLSERVER1
FRMDATADISPLAY MENUSTRIP1 LSTFIELDS FRMDATADISPLAY FRMCONNECT
TABCONTROL1 TABCONTROL1 TABACCESS GBXACCESS4 FRMDATADISPLAY
GROUPBOX1 LSTTABLES FRMDATADISPLAY FRMCONNECT TABCONTROL1
TABORACLE TABACCESS GBXORACLE2 FRMDATADISPLAY GROUPBOX2
LSTVIEWS FRMDATADISPLAY FRMCONNECT TABCONTROL1 TABORACLE
GBXORACLE1 GBXACCESS3 FRMDATADISPLAY GROUPBOX3 LSTFIELDS
FRMDATADISPLAY MENUSTRIP1 39
Exec test count== FRMCONNECT FRMCONNECT FRMCONNECT
FRMCONNECT FRMDATADISPLAY FRMCONNECT FRMCONNECT
FRMDATADISPLAY FRMDATADISPLAY FRMCONNECT FRMCONNECT
FRMDATADISPLAY FRMCONNECT FRMCONNECT FRMDATADISPLAY 15

test list count== FRMDATADISPLAY FRMCONNECT FRMDATADISPLAY
FRMCONNECT TABCONTROL1 TABSQLSERVER GBXSQLSERVER1
FRMDATADISPLAY MENUSTRIP1 LSTFIELDS FRMDATADISPLAY FRMCONNECT
TABCONTROL1 TABCONTROL1 TABACCESS GBXACCESS4 FRMDATADISPLAY
GROUPBOX1 LSTTABLES FRMDATADISPLAY FRMCONNECT TABCONTROL1
TABORACLE TABACCESS GBXORACLE2 FRMDATADISPLAY GROUPBOX2
LSTVIEWS FRMDATADISPLAY FRMCONNECT TABCONTROL1 TABORACLE
GBXORACLE1 FRMDATADISPLAY GBXACCESS3 FRMCONNECT TABCONTROL1
TABSQLSERVER GBXSQLSERVER2 FRMDATADISPLAY FRMCONNECT
TABCONTROL1 TABSQLSERVER GBXSQLSERVER4 FRMDATADISPLAY
FRMDATADISPLAY FRMCONNECT TABCONTROL1 TABSQLSERVER
GBXSQLSERVER5 GROUPBOX3 LSTFIELDS FRMDATADISPLAY MENUSTRIP1 54
Exec test count== FRMCONNECT FRMCONNECT FRMCONNECT
FRMCONNECT FRMDATADISPLAY FRMCONNECT FRMCONNECT
FRMCONNECT FRMCONNECT FRMCONNECT FRMCONNECT FRMCONNECT
FRMDATADISPLAY FRMCONNECT FRMDATADISPLAY FRMCONNECT
FRMCONNECT FRMDATADISPLAY FRMCONNECT FRMCONNECT
FRMDATADISPLAY 21
```

Figure 2. A sample output from the GUI components execution process.

3.3 Using Mutation in Test Case Generation and Execution

The majority of research papers that discussed mutation focused on code, requirement or model mutation. In this part, we will consider test cases' mutation. If a system is expected to accept valid test cases, in principle, it should also reject invalid test cases. This is the main assumption that the paper hypothesis is based on. The GUI model will be tested based on its ability to reject invalid test cases. Mutation is used to make some test cases invalid and test the system ability to reject and catch those mutations.

In specification based mutation, each specification element is replaced with its possible alternatives. A test case that is able to detect the difference between the original specification and the mutated one will kill (i.e. discover) the mutant. Similarly, in code based mutation, a code element (e.g. ">"; the "larger than" symbol) is replaced by one of its possible alternatives (e.g. \geq , $<$, \leq). If none of the test cases in the test suite was able to detect the difference in behavior between the original and mutant code, this means that this mutant is not reachable by any one of the test cases. Another possible reason is that it is possible that the mutant is reachable but shows a similar external behavior relative to the original code.

A test case mutant remains live either:

- Because it is equivalent to the original test case and the application cannot tell the difference between the original behavior and the new one. They could be functionally identical although syntactically different. (i.e. equivalent test cases).
- Or, the program is incapable to kill the mutant. This means that the different behavior is not propagated to the external interface. Those summarize the three conditions to kill a mutant: reachability, infection and propagation.

Test coverage can, therefore, be measured according to the fraction of dead specification or code mutants.

Coverage = Number of mutant test cases discovered / the total number of mutant test cases.

This definition of coverage is somewhat new and is a direct indicator for the GUI model quality. The complete coverage in this approach equals to killing all non-equivalent test cases. Typical coverage types evaluated in testing scope include: requirement, code, statement, branch, path, etc. coverage.

In traditional mutation situation, new test cases are added to kill the mutants. In this approach, this indicates a problem in the GUI model that should be addressed (i.e. why it could not reject an incorrect test case sequence). As this is a model based testing, mutation modifications considered here are only those that are related to the GUI structure. Mutations that are related to the specification such as: invalid user inputs based on boundary values and equivalent partitions are not considered in this research as they will affect the code and not the GUI model.

In this research, the mutation evaluation process is reversed. In the traditional mutation process, the code, or the specification is mutated and the test cases are fixed. It is expected that those test cases can show the difference between the original code or specification and the mutated one. In this research, the code and the specification are fixed and the mutation is occurred in the test cases. It should be mentioned however, that we are not testing the test cases. This type of test case mutation can be classified as a model based testing approach. The GUI model is expected to discover and kill the mutants. As such test adequacy can be measured by the number or the percentage of the failed test cases. Initially, the approach requires calibration to make sure that all original test cases pass (or else take the number of the successful test cases as the denominator). In the first stage before applying test case mutation, all test cases in the suite must be tested to make sure that the GUI model accepts and validates them.

The typical definition of coverage is calculated through the code or specification percentage that is tested through the test cases. In this research, coverage (which is the number of the test cases that fail to the total number of mutated test cases) reflects other quality attributes

in the system. Opposite to this research approach, some good quality attributes of the system such as robustness express the system dynamic range and its ability to tolerate inputs or user mistakes. However, tolerating wrong user inputs should not be mistaken with accepting wrong user inputs. A fault tolerance application may not crash if a user input an invalid input; however, it should reject such input and stop further program execution. This is the quality attribute that this approach is trying to discover in the program; testing its ability to distinguish a correct input from an incorrect one. As the focus of this paper is GUI testing, we will survey some of possible incorrect inputs that an application may experience.

3.4 The automatic execution and verification

Despite the fact that the subject of this paper is test case mutation in GUI models, however the automatic execution and verification process is important in order to evaluate the validity and the value of the proposed mutation operators.

The need for this automatic process was necessary to check whether the GUI model will accept or reject the applied test cases. The main problem was that we are not simply trying to measure expected and actual numeric values which makes the automatic verification process simple. In this approach, there is a need to verify the GUI state before and after executing each test case. Building a research tool to do such tasks may not be easy. There are some known commercial tools such as IBM Rational Robot which may have the capability to do such complex processes. Most commercial tools are using the record/replay methods and few of them use the object data approach that is adapted in this research.

The algorithm we developed to accomplish the automatic execution and verification process depends on using reflection and the fact that managed code includes GUI control details in their executable. The process will use the reverse engineering reflection method to get all GUI controls, their associations and attributes to the memory in order to validate the actual and mutated test cases on them. However,

validating each test case as a one unit was impossible as the process will simply get the GUI controls from the test cases one by one and see if they exist in the GUI or not. The alternative was to consider that if any control fails in a test case, the whole test case will be assumed fail. However, we decided to go with the first option and hence calculate effectiveness based on the controls rather than the test cases. However, the alternative can be later considered to see which approach provides more realistic results.

3.5 The mutation tool

As an extension for GUIAuto [9, 10] which is a test automation tool developed previously by the author, the tool is developed to execute all mutation process activities. The tool first uses reflection (a reverse engineering process to assemble the application components from its executable) to extract all GUI components and their data to an XML file. Other test automation activities such as test case generation, execution and verification can be triggered based on several algorithms and techniques. In the developed tool, GUIAuto is extended based on the GUI mutation operators that will be described later. Those operators can be generated and executed automatically. The focus of the developed mutation tool is on GUI components mutation ignoring the code behind or the actual code in the GUI events triggers.

3.6 The case study

Several small size open source applications are selected for the evaluation in this study. There are two conditions in the selection of those open source codes. The first one is that since the tool uses reflection to serialize .NET managed code and extract all GUI components from it, all selected applications are .NET managed applications. A managed code is a program (in a .NET programming language: C#, VB. Net, managed C++, or JScripts) that is executed within a runtime engine (such as .NET framework and Java Virtual Machine (JVM)) installed in the same machine. The unmanaged code is an executable program that runs as a standalone, launched from the operating system. The

program calls upon and uses the software routines in the operating system. However, it does not require other software application in order to be used.

The second criteria for selection is that the selected application should contain a reasonable amount of GUI components in many forms or web pages in order to construct a GUI hierarchy and be able to generate different sequences of test cases in several levels. Table 1 shows the summary of the 4 selected AUTs for this specific experiment.

Table 1. GUI Summary of the AUTs

AUT	No. of Controls	No. of Paths	No. of Forms
DBSPY	26	19	2
Notepad	158	176	11
BirdWatcher	23	21	3
CourseReg	89	65	2

Based on the GUI model, and based on its structure and components, several mutation operators are proposed. In order to evaluate proposed mutation effects, an automatic execution process is developed. The GUI model takes all test cases as a sequence and the automatic execution process apply those test cases on the actual GUI. An automatic verification method is also developed to check those test cases that are successfully executed. We define execution coverage metric to be the number of executed controls to the number of input or generated controls. The execution process tries to execute the sequence of controls in each test case one by one. The effectiveness for each test case is calculated. The average for the overall test cases is taken to be the execution coverage.

1. Mutation Operator Type 1: Switching GUI components. In this mutation, two components in each test case are switched.

For example, for original test cases of:

1,FRMDATADISPLAY,GROUPBOX3,LSTFIELDS,,SETTINGS

2,FRMDATADISPLAY,MENUSTRIP1,OPENANEWCONNECTIONTOOLSTRIPMENUITEM,,PROGRAM

mutation will be:

```
1,GROUPBOX3,FRMDATADISPLAY,LSTFIELDS,,SETTINGS
2,MENUSTRIP1,ROGRAM,FRMDATADISPLAY,OPENANEWCONNECTIONTOOLSTRIPMENUITEM
```

Note the two controls that are replaced with each other. Table 2 shows test execution effectiveness comparison between original and mutated test cases. All effectiveness metrics calculated below in all tables were based on the number of GUI components successfully executed and verified to the number of the GUI controls that were applied. The number of GUI controls in each test case varies from 3-7 controls in the selected applications. It should be mentioned that earlier we define effectiveness as the number of failed (i.e. detected) mutants to the total number of inputs. This is usually the complement of the effectiveness calculated in the tables below. In Table 2, we didn't focus on the effect of changing the test case generation algorithm as this may not be important to the test case verification processes. This will focus on the impact on the execution processes when test cases are mutated.

Table 2. Execution effectiveness for Type1 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.82
2	50	0.816	0.802
3	30	0.86	0.84
4	50	0.71	0.73

The expectation is that effectiveness for mutants should be less than those of the original test cases. This indicates the application ability to reject wrong test cases. In normal situations, switching elements of the test case should cause a test execution failure. The first type does not imply a failure from the GUI model itself. It implies the inability of the developed execution and verification process to detect this type of mutation. The reason is that the execution process segments each test case in its components and then tries to verify the successful execution of each GUI component individually independent from the

other components in the same test case. The automatic execution and verification process executes and searches for every control from each of the test cases in the application assembly (which contains all application GUI components) and verifies its existence and ability to be executed. This is why switching test case elements didn't affect majorly the effectiveness in Table 1 and made the difference in effectiveness negligible. In reality, the automatic verification process is very complex and subjected to several environmental factors. For example, timing and synchronization between the forms or components that are currently visible is very hard to accomplish. For example, the automatic execution robot maybe expecting a button to be clicked at a moment while the opened form is not yet visible or ready. Another problem is the fact that some modules are modeless and do not accept any further commands before closing. The visibility of some GUI components (especially containers) is also a challenge for the automatic verification where defining its visibilities and executing them can be difficult.

The trials to automatically verify the execution of a complete GUI path were unsuccessful. In future, a modified execution algorithm to verify the test case in the same sequence should be implemented to cover this weakness in the verification process.

2. Mutation Operator Type 2: Changing the name of one control in the sequence (by adding or removing one letter, for example).

In Type 2, a letter from one control in every test case is removed. The goal is to keep the control entity but change its identity. If each GUI object is defined by its name only, this mutation type should be detected. Location of the mutated or modified letter and the control (from the test case) are selected randomly. Table 2 shows the effectiveness results from this mutation gathered from the actual applications. Results showed that execution effectiveness is reduced to indicate reduction percentage for all controls that were located before mutation only. This can be calculated theoretically by:

$$NewEff = OrgEff - (NoMut/NoControlsTotal)$$

Where *NewEff* is the effectiveness after mutation, and *OrgEff*

is the effectiveness before mutation, $NoMut$ is the total number of modified controls (through their name) divided by the total number of controls in all test cases applied.

For example, in Table 2, AUT1, $OrgEff = 0.75$, $NoMut = 30$, and as $NewEff = 0.53$, we can find the total number of controls in all test cases which will be $3000/22$ or about 136 controls (average controls in each test case = $136/30$ or about 4.5). However, this theoretical value is assuming that all mutated controls are undetected and all un-mutated controls are detected in the same way as it was before. Table 3 shows variations of the results between the 4 tested applications.

Table 3. Execution effectiveness for Type 2 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.53
2	50	0.816	0.75
3	30	0.86	0.64
4	50	0.71	0.468

3. **Type 3.** Changing the name of every control in the sequence (by adding or removing one letter, for example). In an extension to type two, and in order to distinguish between a node (i.e. control) failure from a path failure, in this mutation every control name in the test path will be modified by changing only one letter. Table 4 shows the results of applying this mutation.

Table 4. Execution effectiveness for Type 3 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.0
2	50	0.816	0.03
3	30	0.86	0.0
4	50	0.71	0.0

In Table 4, changing all names of controls should bring all test

cases to a complete failure. The few exceptions occur in rare cases where removing a letter from a control change the name to another valid one.

4. **Type 4:** Changing one control from the sequence with another control from the same level. This mutation will bypass some GUI structure constraints where a test case should contain GUI components from the different levels respectively. In some cases, changing this control may change the test case. However, it will produce another valid test case.

Example:

MAINMENU,EDIT,UNDO-TO-FILE,EDIT,COPY

Where Undo and Copy are two controls from the same level. Table 5 shows the results in effectiveness of applying mutation type 4.

Table 5. Execution effectiveness for Type 4 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.73
2	50	0.816	0.815
3	30	0.86	0.81
4	50	0.71	0.82

As expected and explained earlier, switching GUI controls did not affect test case effectiveness as it will modify the test case without invalidating it. In our mutation testing, we are not testing whether the value before and after mutation stays the same. The tests on the GUI model focus on only verifying whether the new mutated test cases will be accepted or rejected by the model. As a result, despite the fact that this type of mutation changes the test case and that the path that it is testing, however, the new test case is a valid one. In some cases as in the last application, effectiveness is improved.

5. **Type 5:** Changing one control in every test case with another one from a different level. In this mutation, one control in each test

case is replaced with a control randomly selected from the pool that contains all AUT controls without observing the location of the newly selected control. Table 6 shows the results from applying this mutation.

Table 6. Execution effectiveness for Type 5 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.64
2	50	0.816	0.812
3	30	0.86	0.80
4	50	0.71	0.82

Similar to Type 1, it is expected that this type should cause a noticeable decrease in effectiveness using mutated test cases. However, this was not the case due to the limitation in the automatic execution and verification process, which verifies the existence of each executed control in the managed code (without considering whether its test case is still valid or not). Rather than lowering the test effectiveness, using mutants improves effectiveness which means that switching the locations of some controls made them more visible and those controls were then located successfully by the execution algorithm.

6. **Type 6:** Deleting a control from a sequence. In this mutation, from each test case, one randomly selected control is removed from the test case.

Table 7. Execution effectiveness for Type 6 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.71
2	50	0.816	0.80
3	30	0.86	0.80
4	50	0.71	0.62

The execution effectiveness should be affected by deleting controls from the test cases solely because of the deletion since the new calcu-

lated effectiveness will be based on the new test cases taken the deletion into consideration. However, all Applications Under Test (AUTs) showed reduction which indicates that when some controls are deleted this may affect the visibility of some other controls.

7. **Type 7:** Adding a control to the sequence. Rather than switching an existed control with another one, in this mutation one randomly selected control is added to each test case. The randomly selected control can be the same added to all or can randomly be selected every time.

Table 8. Execution effectiveness for Type 7 mutation

AUT	No. of test cases	Effectiveness before	Effectiveness after
1	30	0.75	0.76
2	50	0.816	0.77
3	30	0.86	0.77
4	50	0.71	0.75

Similar to the case of: removing a control, adding a control, does not impact effectiveness as this addition is reconsidered when calculating effectiveness. Results showed that although in all mutation cases, test effectiveness after should be less than test effectiveness before, however, as the verification process verifies the controls one by one, the addition of some GUI controls causes the effectiveness to be increased (which may not reflect the test case generation actual effectiveness).

4 Conclusion and Future Work

Test cases are used to detect possible errors and bugs in software applications. In this paper, mutation based testing is used to test applications user interfaces and test if they can differentiate invalid from valid test cases. An automatic tool is developed to automatically generate test cases from applications user interfaces. Later on, and based on the generated test cases, an aspect of one component in each test case

is changed to create test case mutations. Examples of mutations that are considered in this paper were in: changing GUI controls location, name, adding, or removing those controls. An automatic execution and verification process is developed to evaluate the validity of the proposed mutations. The automatic execution and verification processes verify each control individually regardless of its test case. Nonetheless, results showed promising future in the ability of test case mutation to verify certain properties in the GUI model. In mutation original test cases and their results are stored. Those are considered as the baseline for mutation based testing. After generating mutation, to test those mutations, a mutation is said to be killed if its test case result is different from that of the original. The validation of the results considers killing mutants by rejecting them. This makes the automatic verification process difficult due to the difficulty of defining the GUI correct and incorrect states.

References

- [1] Choi, B.J., DeMillo, R.A., Krauser, E.W., Martin, R.J., Mathur, A.P., Offutt, A.J., Pan, H., Spafford, E.H. *The Mothra Tool Set*. Proceedings of the 22nd annual Hawaii international conference on system sciences (HICSS'22), Kailua-Kona, HI , USA, (pp: 275-284), vol. 2 (1989).
- [2] Mateo, P.R. Usaola, M.P. Offutt, J. *Mutation at System and Functional Levels*. Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 6-10, Paris, France, (pp: 110), (2010)
- [3] Offutt, A.J. Jie Pan. *Detecting Equivalent Mutants and the Feasible Path Problem*. Proceedings of the 11th annual conference on computer assurance (COMPASS 96). 17-21 June, Gaithersburg, MD, USA, (pp: 224), (1996).
- [4] Praphamontripong, U. Offutt, J. *Applying Mutation Testing to Web Applications*. Proceedings of the 3rd International Confer-

- ence on Software Testing, Verification, and Validation Workshops (ICSTW), 6-10 April, Paris, France, (pp: 132), (2010).
- [5] DeMillo, R.A. Offutt, A.J. *Constraint-Based Automatic Test Data Generation*. IEEE Transactions On Software Engineering (TOSEM), VOL. 17, Issue 9, (pp: 900), Sep. (1991).
 - [6] Nan, Li Praphamontriping, U. Offutt, J. *An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-uses and Prime Path Coverage*. IProceedings of the International Conference on Software Testing Verification and Validation Workshops, 1-4 April, Denver, CO, USA, (pp: 220), (2009).
 - [7] Offutt, A.J. Rothermel, G. Zapf, C. *An Experimental Evaluation of Selective Mutation*. Proceedings of the 15th International Conference on Software Engineering, 17-21 May, Baltimore, MD, USA, (pp: 100), (1993).
 - [8] Suet Chun Lee Offutt, J. *Generating Test Cases for XML-based Web Component Interactions Using Mutation Analysis*. Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), 27-30 Nov., Hong Kong, (pp: 200-209), (2001).
 - [9] Alsmadi I. Magel K. *GUI Path Oriented Test Generation Algorithms*. Proceedings of Human-Computer Interaction conference (IASTED HCI), Chamonix, France. March 14 - 16, (pp:02), (2007).
 - [10] Alsmadi I. Magel K. *An Object Oriented Framework for User Interface Test Automation*. Proceedings of The Midwest Instruction and Computing Symposium MICS07, 20-21 April, Grand Forks, ND, USA, (2007).
 - [11] Hierons, R.M. Merayo, M.G. *Mutation Testing from Probabilistic Finite State Machines*. Proceedings of the Academic and Industrial Conference – Practice And Research Techniques (TAICPART), 10-14 Sep., Windsor, (pp: 141), (2007).

- [12] Masud, M. Nayak, A. Zaman, M. Bansal, N. *Strategy for Mutation Testing Using Genetic Algorithms*. Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE/CCGEI), 1-4 May, Saskatoon, Canada, (pp 1049- 1052), (2005).
- [13] Bradbury, J.S. Cordy, J.R. Dingel, J. *Comparative Assessment of Testing and Model Checking Using Program Mutation*. Proceedings of the Academic and Industrial Conference – Practice And Research Techniques (TAICPART), 10-14 Sep., Windosr, (pp: 210), (2007).
- [14] Pinto Ferraz Fabbri, S.C. Delamaro, M.E. Maldonado, J.C. Masiero, P.C. *Mutation analysis testing for finite state machines*. Proceedings of the 5th International Symposium on Software Reliability Engineering, (ISSRE), 6-9 Nov., Monterey, CA, USA, (pp: 220-229), (1994).
- [15] Shufang Lee Xiaoying Bai Yinong Chen. *Automatic Mutation Testing and Simulation on OWL-S Specified Web Services*. Proceedings of the 41st Annual Simulation Symposium (ANSS), 13-16 April, Ottawa, CA, (pp: 149), (2008).
- [16] Delamaro, M. Maldonado, J.C. *Interface Mutation: Assessing Testing Quality at Interprocedural Level*. Proceedings of the 19th International Conference of the Chilean Computer Science Society (SCCC), 11-13 Nov., Talca, CHI, (pp:78), (1999).
- [17] Delamaro, M.E. Maidonado, J.C. Mathur, A.P. *Interface Mutation: An Approach for Integration Testing*. IEEE Transactions on Software Engineering (TOSEM), VOL. 27, Issue 3, March, (pp: 228), (2001).
- [18] Serrestou, Y. Beroulle, V. Robach, C. *Functional Verification of RTL Designs driven by Mutation Testing metrics*. Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD), 29-31 Aug., Lubek, (pp: 222), (2007).

- [19] Hoijin Yoon Byoungju Choi Jin-Ok Jeon *Mutation-based Inter-class Testing*. Proceedings of the Asia Pacific Software Engineering Conference (APSEC), 2-4 Dec., Taipei, (pp: 174), (1998).
- [20] Howden, W.E. *Weak Mutation Testing and Completeness of Test Sets*. Transactions on Software Engineering (TOSEM), Vol. SE-8, Issue 4, July, (pp: 371-379), 1982).
- [21] Ferrari, F.C. Maldonado, J.C. Rashid. *A. Mutation Testing for Aspect-Oriented Programs*. Proceedings of the 1st International Conference on Software Testing, Verification, and Validation, 9-11 Apr., Lillehammer, (pp: 52), (2008).
- [22] Gupta, V. *Accelerated GWT: Building Enterprise Google Web Toolkit Applications*. Apress, 1st edition, (2008).
- [23] Bradbury, J.S. Cordy, J.R. Dingel, J. *Mutation operators for concurrent Java (J2SE 5.0)*. Proceedings of the 2nd Workshop on Mutation Analysis (Mutation), 7-10 Nov., Raleigh, NC, USA, (pp: 83-92) (2006).
- [24] R. M. Hierons. *Testing from a finite state machine: Extending invertibility to sequences*. The Computer Journal, 40(4):220-230, (1997).

Izzat Alsmadi,

Received March 5, 2011

Revised April 15, 2012

Izzat Alsmadi
Yarmouk University
Computer Information Systems Department
IT Faculty
Irbid, Jordan
Phone: 9622721111
E-mail: ialsmadi@yu.edu.jo

Annotation on PhD Thesis

Title: Automation of the process of computational linguistic resources creation

Author: Mircea Petic

Place of defence: Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova, Chisinau

Date of defence: 22 January 2012

Speciality: 01.05.01 – Theoretical foundation of computer science; programming

The thesis was elaborated at the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova, Chisinau, in 2011. The thesis is written in Romanian and contains introduction, three chapters, general conclusions and recommendations, bibliography of 200 titles, 14 appendices, 133 pages of the main text, 15 figures, and 44 tables. The results are published in 27 scientific papers.

Keywords: computational linguistic resources, derivational algorithm, affix, prefix, suffix, words segmentation, vocalic/consonantal alternations, automatic derivative generation, generative derivational mechanisms.

The study in this thesis concerns an actual research area related to automation of the process of computational linguistic resources creation, namely, by automatic generation of the derived words that are absent in computational linguistic resources.

The purpose is to study the mechanisms and to elaborate algorithms for automatic generation of the derived words for these resources completion.

The research objectives are: evaluation of the existent methods in the automation of the derivational process; study of the structure particularities of computational linguistic resources available for research; establishing the quantitative and qualitative characteristics of the derived words; elaboration of the algorithms for automatic recognition of the derived words; establishing the mechanisms and elaboration of algorithms for automatic generation of the derived words.

Novelty and scientific originality. This work contributes to complete research in the field of natural language processing by development of mathematical models and algorithms to solve the problem of automatic derivatives

generation. The results of the study represent a realization of a new methodology of studying the issues in computational derivational morphology, related to the algorithmization of certain linguistic mechanisms, such as affixes substitution, derivatives projection, derivational constraints and formal derivational rules.

Important scientific problem solved in the field of research. The problem of automatic generation of derivatives for some classes of Romanian words was solved, which contributes to the facilitation “human – computer” communication in natural language by creation of the computational lexicons, which are the basis of various applications of this field.

Theoretical significance and applied value of the thesis. A statistical method for Romanian affixes uncertainty evaluation based on the notion of entropy was proposed. The mathematical formal descriptions of the derivatives word formation mechanisms were elaborated which served to development of algorithms for automatic generation of the derivatives. During the research the important results were obtained, which permitted to elaborate algorithm for automatic generation of derivatives which can facilitate computational linguistic resources completion. The research results present interest for lexicographic practice, in the process of dictionary elaboration and lexicographic treatment of the derivatives. Also, the results of the investigation can serve as a methodical support in activity of the specialists in both computer science and linguistics.

Implementation of scientific results. An extension of RRTLN database was developed which allowed a correct extraction of about 15.000 derivatives without having a special program of word segmentation in morphemes (41 of prefixes, about 420 of suffixes, over 8 thousand of roots/stems). The established mechanisms, which permitted the elaboration of algorithms and corresponding programs, led to generation a significant number of derivatives with different affixes, 8839 with 11 prefixes, and 2352 with 24 suffixes which will help in Romanian language computational linguistic resources essential enrichment.