STANISŁAW SOSNOWSKI
Polish Academy of Sciences
Laboratory of Computer Modelling
Center of Molecular and Macromolecular Studies
ul. Sienkiewicza 112, 90-363 Lodz, Poland
e-mail: stasosno@cbmm.lodz.pl

Simple and exact Monte Carlo algorithm for modelling of complex polymerization processes*)

Summary — A new, exact Monte Carlo algorithm for the simulation of living/controlled polymerization processes is proposed. In the algorithm macromolecules are represented by doubly linked lists, which are natural models of macromolecules. It allows modelling of elementary reactions by forming/breaking of links in the doubly linked list. As a result the algorithm has a very simple structure and high efficiency. All details of the modeled system: kinetics, molecular weight distributions, microstructure of macromolecules, *etc*., are accessible at any moment of simulation. Practical aspects of implementation of the new algorithm were emphasized: data structures, flowcharts, and source codes (Pascal).
Keywords: computer modelling, kinetics, molecular weight distribution, Monte Carlo simulation, chain microstructure.

PROSTY I DOKŁADNY ALGORYTM MONTE CARLO DO MODELOWANIA ZŁOŻONYCH PROCESÓW POLIMERYZACJI
Streszczenie — W publikacji przedstawiono nowy, dokładny algorytm Monte Carlo do modelowania złożonych procesów polimeryzacji. W algorytmie zrezygnowano z powszechnie stosowanego abstrakcyjnego przedstawiania danych w postaci macierzy i wprowadzono listę podwójnie wiązaną jako naturalny model makrocząsteczek. Dzięki temu reakcje elementarne można modelować jako tworzenie i zrywanie powiązań między elementami listy podwójnie wiązanej. Prowadzi to do prostej struktury oraz dużej szybkości działania algorytmu. W wyniku symulacji są dostępne wszystkie charakterystyki modelowanego procesu: zależności kinetyczne, rozkłady ciężarów cząsteczkowych, mikrostruktura makrocząsteczek itp. W pracy wyeksponowano praktyczne szczegóły implementacji algorytmu: struktury danych, schematy blokowe i kody źródłowe (Pascal). Szczegółowo opisano przykłady algorytmów do symulacji procesów polimeryzacji żyjącej/kontrolowanej, złożonych z różnych reakcji elementarnych: od prostej homopolimeryzacji do kopolimeryzacji obejmującej reakcje inicjowania, propagacji i wymiany segmentalnej.
Słowa kluczowe: modelowanie komputerowe, kinetyka, rozkład ciężarów cząsteczkowych, symulacja Monte Carlo, mikrostruktura łańcucha.

INTRODUCTION

Simulations of polymerization processes could be performed with a number of non-spatial algorithms. However the most often used are variants of the stochastic simulation algorithm (SSA) proposed by Gillespie. SSA, as well as other algorithms described in the paper, is valid when diffusion is much faster than any reaction in the system [1].

In the SSA each step of simulation consists of several stages: first a vector R is constructed containing n stochastic rates R_i of n reactions possible for a given set of reactants. A stochastic reaction rate is a product of a "chemical" rate constant, a coefficient dependent on the reaction order, and the number of all possible combinations of chemical entities involved in the given reaction. Then, a reaction μ is selected using a uniform random number r_i ($0 \le r_i < 1$), which satisfies condition (1):

$$\sum_{i=1}^{\mu-1} R_i < r_1 \cdot \sum_{i=1}^{n} R_i \le \sum_{i=1}^{\mu} R_i \qquad (1)$$

*) Materiał przedstawiony w tej publikacji był prezentowany jako wykład sekcyjny podczas Zjazdu PTCh i SITPChem w Gliwicach, w dniach 14—18.09.2010 r.

Next, a time interval Δt, in which the reaction will occur, is calculated from equation (2):

$$\Delta t = \frac{-\ln(r_2)}{\sum_{i=1}^{n} R_i} \tag{2}$$

Where r_2 is a uniform random number from the range (0,1). Finally, numbers of reactants are updated according to the processed reaction.

One of the most characteristic features of the polymerization processes is a large number of reactants. Formally, each chain of discrete length constitutes a distinct reactant in the sense of the SSA, thus the number of reactions n increases with broadening of molecular weight distribution (*MWD*). The number could increase further when redistribution of active centers is allowed. Even worse case is copolymerization, in which practically each chain constitutes a separate chemical entity because of additional differentiation caused by distinct arrangement of constitutional units along a chain. Hence, the reaction number n could be close to the number of modeled chains. A modification of the original Gillespie algorithm was proposed in order to limit the number of distinguished reactions [2]. On the other hand, variants of SSA with a random selection of a reacting chain from a population (instead of selection of a reaction) are not consistent with the Gillespie idea and, as it was shown by Szymanski, could lead to erroneous results [3].

Beside numerous variants of the Gillespie algorithm, two other algorithms worth mentioning were published. They were designed strictly for modelling of polymerization processes. Hamaide presented a very simple algorithm for ionic and radical copolymerizations, which allows modelling of *MWD* and chains microstructure as a function of conversion, but not in the time domain [4]. An algorithm proposed by Szymanski permits modelling in the same range as SSA. In the algorithm, chains are selected sequentially and are subjected to reaction within an arbitrary chosen time interval ΔT, within which the concentration of reagents is assumed to be approximately constant. During that time, the selected chain may react several times. Selection of a distinct reaction (*e.g.*, propagation, transfer, etc.) and calculation of a time interval Δt required for the reaction are performed similarly to the original Gillespie algorithm. When the sum of Δt for the given chain exceeds ΔT then a next chain is selected. Concentrations of reagents are updated after transformation of the last chain and the next turn begins [3, 5].

A new algorithm was developed by Sosnowski for modelling of copolymerization processes consisting of initiation and propagation reactions. The algorithm surpasses both above ones in respect of exactness and (in specific applications) also in speed [6]. It will be described in detail later in this work.

An important aspect of any algorithm is the structure of data representing reacting substrates and products. In this respect arrays have dominated nearly completely all other useful data structures. The only exception is a binary tree and a singly linked list applied in singular systems [2, 7]. It will be shown in this paper, that representation of data by a doubly linked list allows a significant advance in modelling of complex polymerization processes.

SIMULATION OF ELAPSE OF TIME

This is a central point in modelling of reactions kinetics. In SSA the time interval necessary to accomplish an elementary reaction is inversely proportional to the sum of stochastic rates [see equation (3)]. Because stochastic rate constants are invariant then the length of the interval depends on current concentrations of reagents. Exactly one elementary reaction proceeds within the interval. The selected reaction determines only types of reactants involved, not individual molecules. Therefore the algorithm can be defined as oriented toward the reaction.

On the contrary, in Szymanski algorithm each chain is subjected to react several times within a short, arbitrarily chosen, time interval. A number of elementary reactions allowed for a selected chain depends on its properties (length, composition, reactivity etc.), and on concentrations and reactivities of other components present in the mixture. More precisely, the number is related to the time calculated analogously to SSA [equation (2)], but only reactions possible for the selected chain are accounted. Thus the algorithm is oriented toward chains (reactants). Modelling of microstructure, topology, and *MWD* is therefore more straightforward than in SSA. However the selected length of the time interval is a compromise between speed and accuracy of a simulation.

In the algorithm proposed here elapse of time is also measured using an interval with a constant length. It will be shown, that this approach allows simplifying the structure of algorithms. In opposition to Szymanski algorithm, the length is not arbitrarily selected, but is deduced from characteristics of the modelled system [6]. Appropriate equations for selected examples will be derived later. Within the interval an individual active center and an individual monomeric unit are randomly selected from populations presented in the simulated system. Types of the selected molecules determine possible reactions (initiation, propagation, chain transfer, and so on), which are randomly permitted or not, depending on values of suitable rate constants. Thus, the algorithm is oriented toward reactants, what will be also emphasized by data structures used for representation of substrates and products.

DATA STRUCTURES

According to Niklaus Wirth opinion, preselection of a data structure precedes all other programming tasks. Moreover, the chosen data structure forces a programmer to use an appropriate, less or more efficient, algorithm

[8]. It also has been shown that proper selection of data structures allows writing simple and efficient programs. However optimal structures differ from one problem to another [9]. Fortunately, there are many different structures which can be assorted with specific needs. Some examples how they can be applied to specific polymerization processes will be presented below.

The simplest case — modelling of monomer conversion in a living/controlled homopolymerization with fast initiation

We are interested in changes of the number of molecules of only one reagent. The changes are described by the following differential equation:

$$\frac{\mathrm{d}[M]}{[M]} = -k_p[I]_0\,\mathrm{d}t \tag{3}$$

For the elementary change of the number of monomer molecules, the equation (3) can be approximated by the discrete form:

$$\frac{1}{n_M} = \frac{k_p n_{0I}}{VN_A}\Delta t \tag{4}$$

where: *V — volume of polymerization mixture, N_A — the Avogadro constant, [M], n_M — concentration and number of monomer molecules, respectively, $[I]_0$, n_{0I} — concentration and number of initiator molecules, respectively, Δt — a length of time interval necessary to decrease the number of monomer molecules by one, k_p — the propagation constant.*

The length of time interval Δt_0, sufficient for the first act of propagation to occur (on average) can be calculated from equation (4) after substitution of the initial number of monomer molecules n_{0M}.

$$\Delta t_0 = \frac{1}{n_{0M}} \cdot \frac{VN_A}{k_p n_{0I}} \tag{5}$$

In other words, the first act of propagation is a certain event within the time Δt_0. Other acts of propagation, when a number of monomer molecules is lower than the initial one, take longer times, defined by equation (4). Hence, at any moment of the process, probability p that the next act of propagation will happen within the time interval Δt_0 is given by the ratio $p = \Delta t_0/\Delta t = n_M/n_{0M}$ [6].

We need to follow changes of the number of monomer molecules within time. Because all monomer molecules are identical, we can use a single variable M ($M \equiv n_M$) for storing the number. The simplest structure which has all necessary properties is a nonnegative (unsigned) integer. However for better efficiency one should use an integer type native for the selected programming language (or targeted processor). The initial value of the counter is equal to the number of monomer molecules at the start of polymerization ($M_0 \equiv n_{0M}$). The simulation should stop when the value of the M counter reaches zero or it equals an arbitrary chosen number. Elapse of time during the simulation can be followed by counting iterations, so a second variable It of integer type is necessary. Variable dt of real type stores the value of the time interval Δt_0.

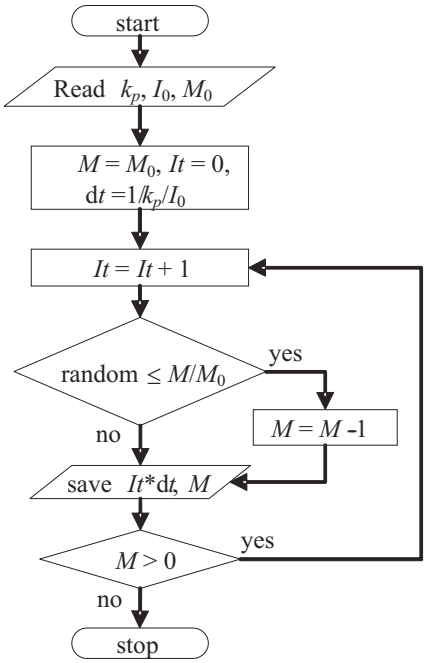

Fig. 1. Flowchart for modelling of monomer conversion in a living/controlled homopolymerization with fast initiation

The resulting algorithm (Fig. 1) and the source code of resulting program (in Pascal, shown below) are very simple and do not need comments. The program is valid for all initial values. For sake of clarity: the logical condition random ≤M/M_0 from the flowchart was transformed into random (M_0) < M used in source code due to some properties of statements specific to Pascal language — both formulations are strictly equivalent.

```
var
  kp, Io, dt : Real;
  It, M, Mo : Integer;
begin
  ReadLn(kp);      // propagation rate constant
  ReadLn(Io);      // Initiator concentration
  ReadLn(Mo);      // high value improves precision
  M := Mo;
  It := 0;
  dt := 1.0/kp/Io;
  repeat
    Inc(It);
    if Random(Mo) < M then Dec(M);
    WriteLn(It*dt, M);
  until M = 0;
end;
```

Figure 2 shows that, despite of simplicity of the algorithm, the Monte Carlo method and analytical solution yield consistent results. It should be stressed that the plot was obtained for the only single run with only one hundred of monomer molecules. The execution time measured by system clock was bellow one millisecond on 3.0 GHz Core^{TM}2 Duo machine. For a better approximation of the

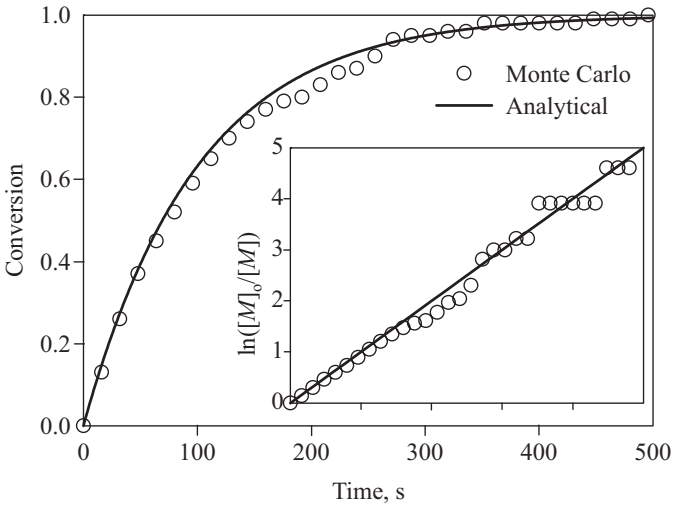

Fig. 2. An example of simulations of monomer conversion in a living/controlled homopolymerization with fast initiation. Conditions: $n_{0M} = 100$, $k_p = 1.0\ l \cdot mol^{-1} \cdot s^{-1}$, $[M]_0 = 1.0\ mol \cdot l^{-1}$, and $[I]_0 = 0.01\ mol \cdot l^{-1}$. The bottom axis for the semilogarithmic plot has the same range as that for the conversion plot

analytical solutions it is recommended to average results of several simulations or to increase the scale of modelling by substitution a large value for M_0.

Intermediate case — modelling of monomer conversion and *MWD* in the living/controlled homopolymerization with slow initiation

Changes of reagents concentration in the living/controlled homopolymerization with slow initiation are described by the following set of differential equations:

$$\frac{d[I]}{dt} = -k_I[I][M]$$
$$\frac{d[M]}{dt} = -k_I[I][M] - k_p([I]_0 - [I])[M] \quad (6)$$

and *MWD* is given by equation derived by Gold [9]:

$$N(r) = \frac{1}{[I]_0 - [I]} \cdot \frac{[I]}{\gamma}\left(\frac{\gamma}{\gamma - 1}\right)^r \left[1 - \left(\frac{[I]}{[I]_0}\right)^{\gamma-1} \sum_{S=0}^{r-1} \frac{\left[(\gamma - 1)\ln\frac{[I]_0}{[I]}\right]^S}{S!}\right] \quad (7)$$

where: γ — *the ratio of the propagation rate constant to the initiation rate constant*.

Concentrations of substrates can be obtained by numerical integration of equations (6) and *MWD* from relatively simple calculations from equation (7). It should be noted, that for calculation of *MWD* we need an instant value of initiator concentration. However the above approach yields only approximate results due to errors specific for calculations on finite precision numbers [10]. Moreover, some calculations (*e.g.* a factorial) are limited by the selected type of variables used for storing data. For example Pascal allows calculations of factorial for arguments up to 170 for real type (8 bytes long) and up to 1754 using extended type (10 bytes long). On the other hand, the proposed Monte Carlo algorithm is free from such restrictions and other errors typical for numerical solutions.

In the considered case a variant of the algorithm is proposed, more general than necessary. It allows modelling of polymerization, in which monomer molecule attachment rate constant depends on the degree of polymerization (*DP*) of the reacting chain. In our particular case rate constants are equal for all chain lengths, apart of the zero length, which is ascribed to an initiator molecule. Such a general approach leads to the simplest structure of the algorithm.

The first elementary reaction will occur in the system (on average) after the time interval Δt_0 calculated from equation (4) for the initial numbers of reactants and the highest rate constant [6]. All other reactions need a longer time. Due to constant number of active centers, probability of each elementary reaction is expressed by equation (8):

$$p = \frac{\Delta t_0}{\Delta t} = \frac{n_M}{n_{0M}} \cdot \frac{k_{DP}}{\max(k_{DP})} = \frac{n_M}{n_{0M}} r_{DP} \quad (8)$$

where: r_{DP} — *the relative reactivity of an active center situated on a chain with* $DP \geq 0$.

Relative reactivities r_{DP} are calculated from rate constants k_{DP} prior to simulation and are stored in array *R* containing variables of real type and index range from 0 to maximal *DP* allowed. Chain lengths are stored in array

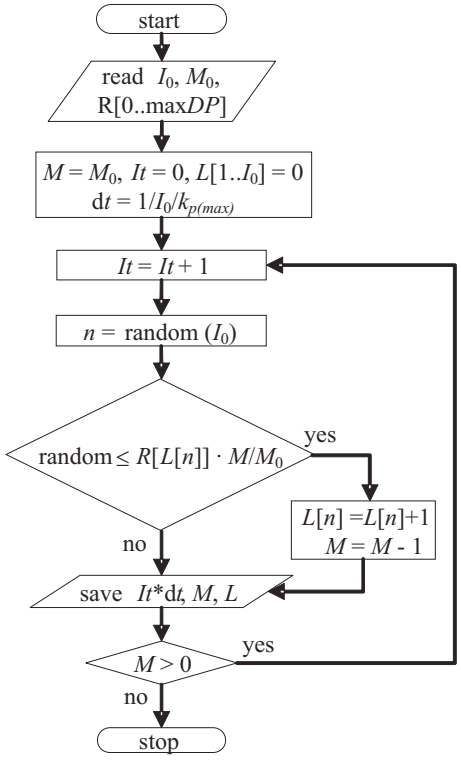

Fig. 3. Flowchart for modelling of monomer conversion and MWD in a living/controlled homopolymerization with slow initiation

L of integer type, indexed from 1 to the number of chains $I_0 \equiv n_{0I}$. Variables *It*, I_0, *M*, M_0, and d*t* were used like in the previous example.

A flowchart of the discussed algorithm is shown in Figure 3. The iteration loop consists of very few statements. Firstly, the iteration counter is increased. Then a chain is selected and its length is read from array *L*. The relative reactivity, respective to the length, is extracted from array *R*. Afterwards the logical condition is evaluated and, if it is true, length of the selected chain is incremented and number of monomer molecules is decremented. Output data can be saved before the next iteration starts. The output data allow to calculate monomer conversion and *MWD* after each step of propagation (if necessary), or once after finishing the simulation. The Pascal code of the main loop is presented below.

```
repeat
  Inc(It);
  n := Random (Io);
  if Random (Mo) < M*R[L[n]] then
    begin
      Dec(M);
      Inc(L[n]);
    end;
  WriteLn(lt*dt, M, L);
until M = 0;
```

Analogously to the former example, a small difference between formulation of logical expressions from the flowchart and from the Pascal code is caused by peculiar character of Pascal statements.

Results of modelling of the system are shown in Fig. 4 and Fig. 5. All plots in these figures confirm excellent consistency of data obtained from the Monte Carlo simulations and from analytical solution or integrations of ordinary differential equations [equation (6)]. As it was indicated earlier, not all *MWD* can be calculated using the Gold equation (7). When the ratio of the initiation rate constant to the propagation rate constant is high (*e.g.* 0.2, the dashed plot in Fig. 5) then all initiator is consumed at relatively low conversion of monomer. In consequence, expression $[I]_0/[I]$ became undefined and equation (7) cannot be applied after the moment of full consumption of initiator. In the discussed case the plot was obtained using the smallest possible, but finite, value of [*I*]. The time of the simulation of 10^5 chains was about 3 s.

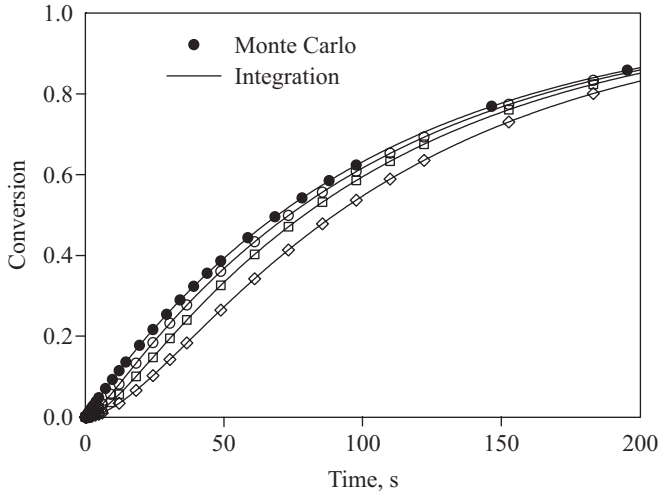

Fig. 4. Modelling of monomer conversion in a living/controlled homopolymerization with slow initiation. Conditions: n_{0M} = $1 \cdot 10^7$, $n_{0I} = 1 \cdot 10^5$, $[M]_0 = 1.0$ mol · l^{-1}, $[I]_0 = 0.01$ mol · l^{-1}, and k_p = 1.0 l · mol^{-1} · s^{-1}. k_I: (●) 1.0, (○) 0.2, (□) 0.1, (◊) 0.5 l · mol^{-1} · s^{-1}, respectively

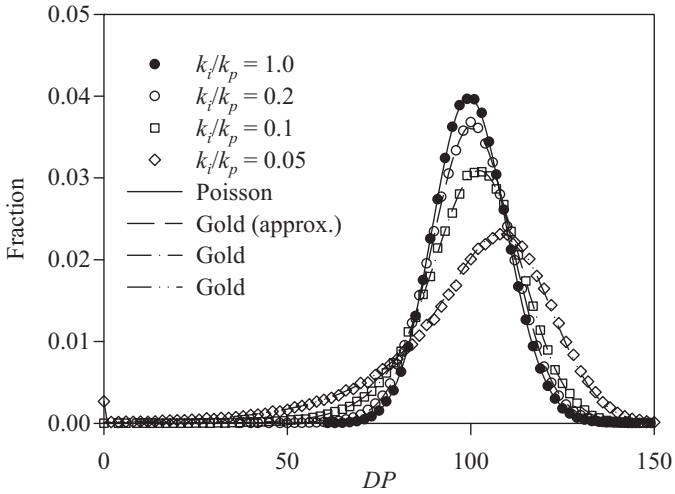

Fig. 5. Modelling of MWD in the living/controlled homopolymerization with slow initiation. Conditions: $n_{0M} = 1 \cdot 107$, n_{0I} = $1 \cdot 10^5$, $[M]_0 = 1.0$ mol · l^{-1}, $[I]_0 = 0.01$ mol · l^{-1}, and k_p = 1.0 l · mol^{-1} · s^{-1} · k_I: (●) 1.0, (○) 0.2, (□) 0.1, (◊) 0.5 l · mol^{-1} · s^{-1}, respectively

Complex case — modelling of monomer conversion, *MWD*, and microstructure of chains in copolymerization with segmental exchange

This is the case when propagation reaction is accompanied with an intra- and intermolecular chain transfer reactions resulting in an exchange of segments between reacting chains and in reordering of constitutional units. A well-known example is a sequential copolymerization of heterocyclic monomers yielding less or more random (instead diblock) copolymers [12]. The segmental exchange reactions have been modeled extensively in case of homopolymerization [5, 13]. However, in the case of copolymerization they were not yet modeled numerically.

The process discussed here consists of the following reactions: initiation, propagation and intermolecular chain transfer. Intramolecular chain transfer reactions (back-biting) will be omitted in this paper for simplicity. All of mentioned reactions are visualized in equations (9).

$$
\begin{aligned}
&I^* + A \xrightarrow{k_{(I^*,A)}} IA^* \\
&I^* + B \xrightarrow{k_{(I^*,B)}} IB^* \\
&\sim A^* + A \xrightarrow{k_{(A^*,A)}} \sim AA^* \\
&\sim A^* + B \xrightarrow{k_{(A^*,B)}} \sim AB^* \\
&\sim B^* + A \xrightarrow{k_{(B^*,A)}} \sim BA^* \\
&\sim B^* + B \xrightarrow{k_{(B^*,B)}} \sim BB^* \\
&I^* + X\mathbf{A}M_n^* \xrightarrow{k_{(I^*,Ac)}} I\mathbf{A}M_n^* + X^* \\
&I^* + X\mathbf{B}M_n^* \xrightarrow{k_{(I^*,Ac)}} I\mathbf{B}M_n^* + X^* \\
&\sim M_mA^* + X\mathbf{A}M_n^* \xrightarrow{k_{(A^*,Ac)}} \sim M_mA\mathbf{A}M_n^* + X^* \\
&\sim M_mA^* + X\mathbf{B}M_n^* \xrightarrow{k_{(A^*,Bc)}} \sim M_mA\mathbf{B}M_n^* + X^* \\
&\sim M_mB^* + X\mathbf{A}M_n^* \xrightarrow{k_{(B^*,Ac)}} \sim M_mB\mathbf{A}M_n^* + X^* \\
&\sim M_mB^* + X\mathbf{B}M_n^* \xrightarrow{k_{(B^*,Bc)}} \sim M_mB\mathbf{B}M_n^* + X^*
\end{aligned}
\qquad (9)
$$

where: *I, A, B — designate initiator and monomer molecules, respectively, X* — stand for a chain fragment with active centers I*, A*, or B*. Bolded characters mark monomeric units being an object of the attack.*

In this example we are interested in the microstructure of macromolecules, thus we have to analyze the microstructure as the simulation occurs or to store data for latter processing. The second approach yields a simple and fast algorithm and allows preparation of a "product sample", which can be analyzed in every respect at any time. Some issues of the algorithm (*e.g.* elapse of time and selection of probability of reaction) were described in details earlier [6]. In this paper a new manner of data representation will be introduced.

The microstructure of chains is usually mapped by a regular array containing simple variables, for example integers or characters, representing singular monomeric units. Each row (or column) of the array represents one chain. In such case the size of the array cannot be predetermined, because it is not possible to define an upper limit of *DP* of chains. Hence, it is necessary to use an excessive amount of memory (above 20-fold of the mean *DP*) for each chain, but only a strict control of length of chains can prevent errors of range overflow. An alternative of arrays are doubly linked lists [14]. They are much better suited to the problem discussed here. The first advantage is that the amount of the needed memory can be exactly foreseen *a priori*, before the start of simulation. Next, they require only *c.a.* three-fold excess of memory (in respect to the number of monomeric units). Finally, as it will be shown, all reactions giving involved chains can be modeled straightforwardly.

Lets assume that each molecule has a few fundamental properties: a type [A, B, or I in the sense explained below the equation (9)], a host chain (to which it was attached), and two indicators (pointers) to molecules which it was connected to. The indicators play a role of functionality of real molecules. It is reasonable to introduce further types of molecules, namely Ac and Bc for monomeric units built into chain, and I*, Ac*, and Bc* for molecules being an active center. All mentioned properties could be represented by a record structure, allowed in most of programming languages. Records can be linked together into a doubly linked list, which resembles a real macromolecule. Appropriate structures are shown in Fig. 6. The figure presents a several randomly selected molecules (identified by numbers 173, 13, 777, 55, for example) arranged into a doubly linked list. Fields "next" and "previous" of each record contain identifiers of the attached molecules. For an initiator and active centers one of these fields contains a pointer to a non-existing molecule (*nil*). By convention used in algorithmics, an arrow going out from a given field indicates a molecule represented by the identifier stored in the field.

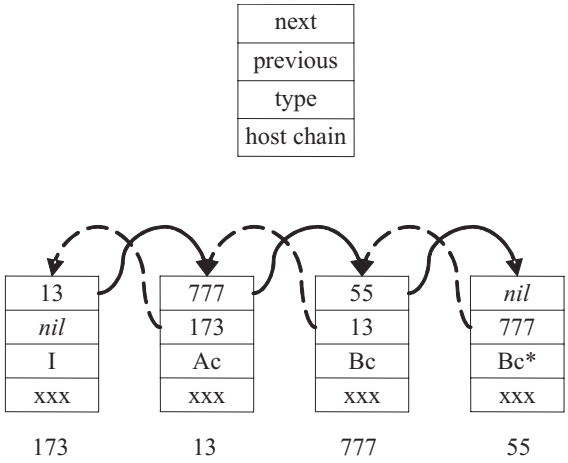

Fig. 6. Structures used for representation of molecules. See description in the text

In a genuine reaction a casual macromolecule (more exactly an active center) reacts with a random monomeric unit. Thus, we need a method of random selection of both entities from populations. This is possible by using three arrays. Array *M* permanently stores pointers to all monomeric units and enables their random selection. Array *C* permanently stores pointers to all initiator molecules and provides an access to beginning of every chain. Array of pointers *AC* allows random selection of an active center. Initially it contains the same data as array *C* but its content is modified after each elementary reaction which changes placement of active centers.

A model of segment exchange reaction by intermolecular chain transfer is schematically illustrated in Fig. 7. It consists of few stages and is performed by simple exchange of data between records. At the beginning a generated random number allows selection of a pointer from array *AC*. This pointer indicates, therefore, a random active center. Analogously, a second random number yields an index of a pointer from array *M*, which pro-

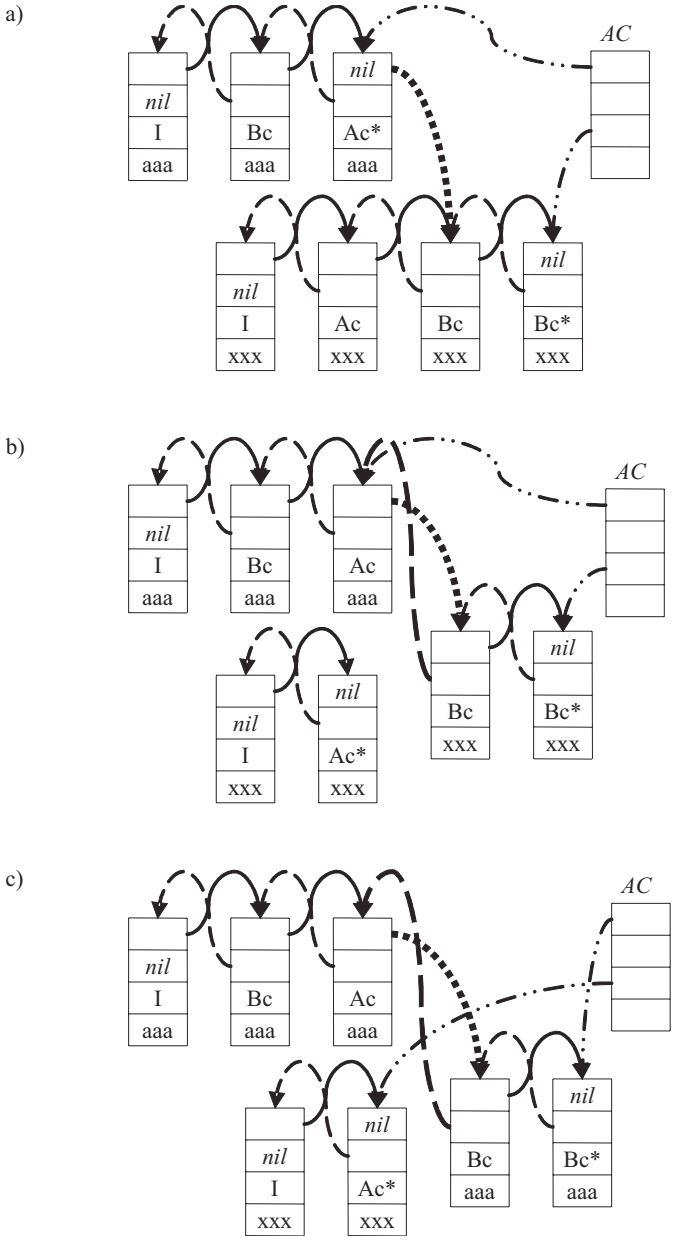

Fig. 7. Illustration of process of segment exchange between macromolecules: a) — attack of an active center on a monomeric unit randomly selected from the population, b) — breaking of bonds between the attacked monomeric unit and its predecessor, c) — connecting of attacked unit to a new host chain and updating pointers to new active centers

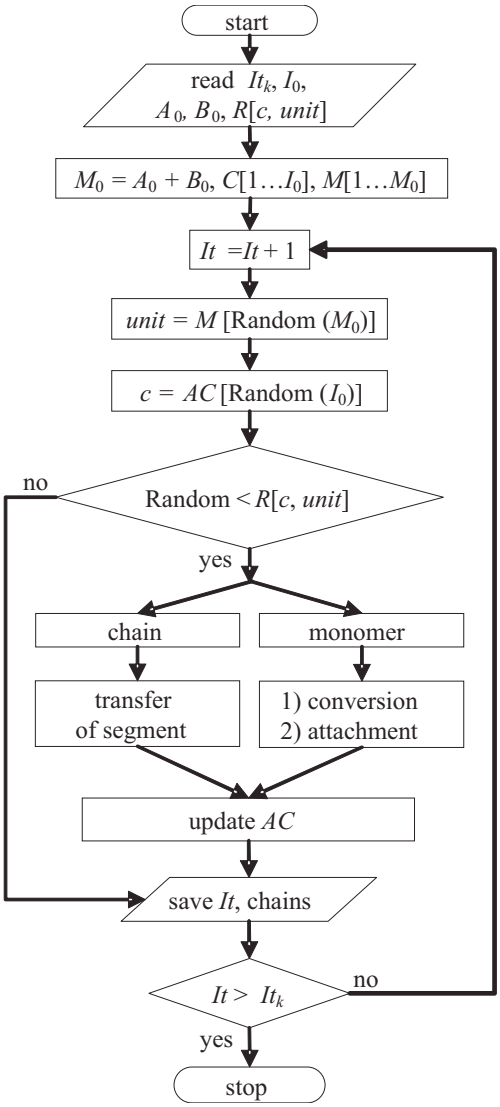

Fig. 8. Flowchart for modelling of monomer conversion, MWD, and microstructure of chains in copolymerization with segmental exchange

vides access to an attacked monomeric unit. When the attacked monomeric unit is a monomer molecule then propagation occurs. When the unit is built in a chain, then it is decided automatically which chain is undergoing the reaction. It is worth to note, that selection of an active center provides access to each singular macromolecule with an equal probability, but selection of a monomeric unit allows selection of a macromolecule with probability proportional to its *DP*. Moreover, the attacked monomeric unit ensures access to any monomeric unit in its host chain by appropriate pointers stored in the record representing the unit. After executing of the stage a), the field "next" in the record representing the active center indicates the attacked unit.

In the next stage (see Fig. 7b) the bond between the attacked unit and its predecessor in the host chain is broken (*e.g.* the field "next" in the predecessor record was set to *nil*). Furthermore the field "previous" in the record representing the attacked monomeric unit points to the active center involved in the reaction. Finally, in the last stage (Fig. 7c), the number of the attacking chain is assigned to "host" property in all records representing transferred monomeric units, and new placements of active centers are stored in *AC* array.

All presented above operations are illustrated by the flowchart in Fig. 8 and a fragment of source code. In the source code types of monomeric units were replaced by numbers to improve efficiency. Hence, conversion of a monomer molecule (designated by 2 and 3 for monomers A and B, respectively) into a monomeric unit in a chain (denoted as 0 or 1) can be realized by command

Dec(*TheMerType*, 2). A variable *UnknownMer* is the sentinel facilitated control the internal loop.

```
repeat
Inc(i);
TheMer := M [Random(MaxMonomerNumber)];
TheMerType := TheMer.MerType;
TheCenterNumber := Random(MaxChainsNumber);
TheCenter := AC[TheCenterNumber];
TheCenterType := TheCenter^.MerType;
if Random >= Rate[TheCenterType, TheMerType]
then Continue;
case TheMerType of
2, 3: begin                          //propagation
   Dec(TheMerType, 2);      //conversion of unit type
   TheMer^.Previous := TheCenter;
   TheMer^.MerType := TheMerType;
   TheMer^.HostChain := TheCenter^.HostChain;
   TheCenter^.Next := TheMer;
   AC[TheCenterNumber] := TheMer;
   end;
0,1: begin                          //reshuffling
   AttackedChain := TheMer.HostChain;
   if TheCenter^.HostChain = AttackedChain
   then Continue;
//change of host property in each transferred unit
   ThePreviousMer := TheMer.Previous;
   ThePreviousMer^.Next := UnknownMer;
   CurrentMer := TheMer;
   while CurrentMer  UnknownMer do begin
      CurrentMer^.HostChain := TheCenterNumber;
      CurrentMer := CurrentMer.Next;
      end;
//updating information in AC array
   TheMer^.Previous := TheCenter;
   TheCenter^.Next := TheMer;
   AC[TheCenterNumber] := AC[AttackedChain];
   AC[AttackedChain] := ThePreviousMer;
   end;    //MerType 0,1
end;    //case MerType of
until i > ik;
```

As it can be seen, there are no arithmetic calculations in the main loop. All operations involve only generation of random numbers, retrieving of pre-calculated rate constants, and transfer of data from one record to another.

The algorithm was verified on data published by Szymanski for a process of homopolymerization with reshuffling [5]. This process can be modeled using the proposed algorithm by simply assuming the same reactivities of both comonomers and of both kinds of monomeric units in chains, *e.g.* $k_p = k_{(I^*,A)} = k_{(A^*,A)} = k_{(B^*,A)} = k_{(I^*,B)} = k_{(A^*,B)} = k_{(B^*,B)}$ and $k_{tr} = k_{(I^*,Ac)} = k_{(A^*,Ac)} = k_{(B^*,Ac)} = k_{(I^*,Bc)} = k_{(A^*,Bc)} = k_{(B^*,Bc)}$. Other parameters applied in this work are: $n_{0M} = 1 \cdot 10^8$, $n_{0I} = 1 \cdot 10^6$, $[M]_0 = 1.0\ \text{mol} \cdot \text{l}^{-1}$, $[I]_0 = 0.01\ \text{mol} \cdot \text{l}^{-1}$, $k_p = 1.0\ \text{l} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$, and $k_{tr} = 0.01\ \text{l} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$. Final monomer conversion = 80 %. The time of the simulation was about 200 s.

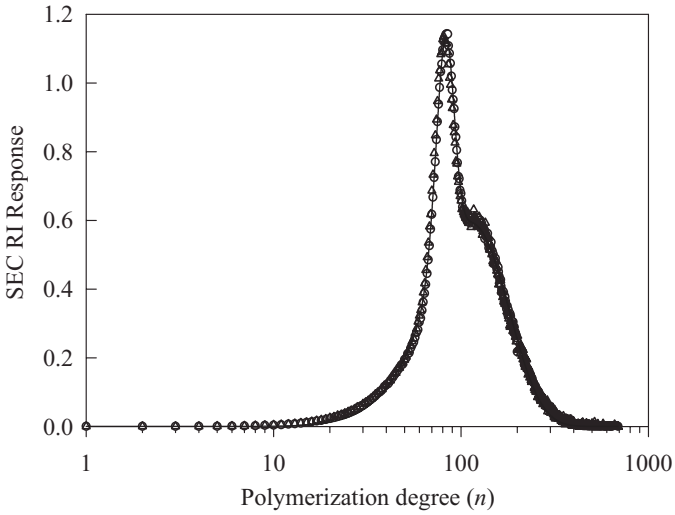

Fig. 9. Comparison of MWD obtained from Monte Carlo simulations and integration of differential equations. Triangles — the new Monte Carlo algorithm; circles — Monte Carlo algorithm proposed by Szymanski; line — integration of differential equations. Conditions described in the text

Figure 9 shows that *MWD* obtained from the algorithm does not differ either from that, produced by Szymanski method, or from that obtained by numerical integration of (approximated) differential equations.

CONCLUSIONS

The new algorithm allows straightforward modelling of kinetics of polymerization processes. With proper structures used for representation of data, it appears to be a convenient tool for modelling of processes with various degree of complexity. These structures imitate properties of genuine molecules, so reactions can be simulated by breaking and creating links between representing structures, analogously to natural behavior of molecules. Due to elimination of arithmetic calculations, many possible sources of errors can be avoided, providing a compact and safe program code. Despite of simplicity the algorithm allows accurate prediction of monomer conversion, *MWD*, and microstructure of macromolecules simultaneously in a single simulation.

ACKNOWLEDGMENTS

The author expresses his gratitude to Prof. Ryszard Szymanski for data used in Fig. 9. This work was financially supported by The Polish Ministry of Science and Higher Education from budget funds for science for years 2009—2012.

REFERENCES

1. Gillespie D. T.: *J. Phys. Chem.* 1977, **81**, 2340.
2. Chaffey-Millar H., Stewart D., Chakravarty M. M. T., Keller G.: *Macromol. Theory Simul.* 2007, **16**, 575.
3. Szymanski R.: *e-Polymers* 2009, 044.

4. (a) Hamaide T.: *Actual Chimique* 1997, **5**, 29. (b) Hamaide T.: *Actual Chimique* 1997, **10**, 26.
5. Szymanski R.: *Macromol. Theory Simul.* 1998, **7**, 27.
6. Sosnowski S.: *Macromol. Theory Simul.* 2010, **19**, 483.
7. Hendrickson R. C., Gupta A. M., Macosko C. W.: *Comput. Polymer Sci.* 1994, **4**, 53.
8. (a) Wirth N.: "Algorithms + Data Structures = Programs", Prentice Hall, 1976. (b) Wirth N.: "Algorytmy + struktury danych = programy", Wyd 6., WNT, Warszawa 2002.
9. Bentley J.: "Writing efficient programs", Prentice Hall, Englewood Cliffs 1982, Ch. 4.
10. Gold J.: *J. Chem. Phys.* 1958, **28**, 91.
11. (a) "Accuracy and reliability in scientific computing" (Ed. Einarsson B.), SIAM, Philadelphia 2005. (b) Higham N. J.: "Accuracy and stability of numerical algorithms", 2nd Ed., SIAM, Philadelphia 1996. (c) Overton M. L.: "Numerical Computing with IEEE Floating Point Arithmetic", SIAM, Philadelphia 2001. (d) Muller J. M. *et al*.: "Handbook of Floating-Point Arithmetic", Birkhäuser, Boston 2010.
12. (a) Kasperczyk J.: *Macromol. Chem. Phys.* 1999, **200**, 903. (b) Florczak M., Libiszowski J., Mosnacek J., Duda A.: *Macromol. Rapid Commun.* 2007, **28**, 1385.
13. (a) Szymanski R., Baran J.: *Macromol. Theory Simul.* 2002, **11**, 836. (b) Szymanski R.: *Macromolecules* 2002, **35**, 8239. (c) Szymanski R., Baran J.: *Polimery* 2003, **48**, 758.
14. (a) Aho A. V., Hopcroft J. E., Ullman J. D.: "Algorytmy i struktury danych", Helion, Gliwice 2003. (b) Aho A. V., Hopcroft J. E., Ullman J. D.: "Data structures and algorithms", Addison-Wesley, London 1983.