Automatic Droplet Sequence Generation for Microfluidic Networks with Passive Droplet Routing

Andreas Grimmer *Student Member, IEEE,*
Werner Haselmayr *Member, IEEE,* and Robert Wille *Senior Member, IEEE*

***Abstract*—Droplet-based microfluidic devices are a well-established and highly potential *Labs-on-Chip* (LoC) technology as droplets are especially suited to encapsulate biological samples like cells, proteins, or DNA. These droplets are injected in a continuous phase and flow through closed microchannels to modules executing operations on the droplets – eventually realizing a (bio-)chemical experiment. Moreover, this technology even allows for the realization of multiple experiments on a single device by letting droplets take different paths through the microfluidic network. This requires, however, a mechanism to route the droplets along these paths. To this end, the concept of *passive droplet routing* has been suggested which entirely avoids complex valves or switches and, instead, realizes the routing by exploiting the hydrodynamic effect that a droplet will always flow along the path with the highest volumetric flow rate. Since droplets themselves affect the volumetric flow rate, a dedicated sequence of droplets can define what path is taken and, hence, what experiment is executed. However, determining such a droplet sequence is a non-trivial task, as it is non-obvious how much droplets are needed, when to inject them, and how they are interacting. In this work, we are addressing this issue by providing, for the first time, an automatic method for the generation of droplet sequences realizing the desired experiments on a given network. Evaluations confirm the practicability of the proposed solution. Moreover, the suitability of the obtained droplet sequences is additionally validated through simulations on the 1D analysis model.**

***Index Terms*—Droplet microfluidics, microfluidic networks, droplet routing, design automation.**

I. Introduction

Microfluidics allow to miniaturize the classical test tube by using droplets of nano- to femto-liter volumes [1]. For these *droplet-based* microfluidics, two main setups got established in the past: In *two-phase flow* microfluidics, the droplets (i.e. the first phase) flow in closed channels inside an immiscible continuous phase (i.e. the second phase) which is driven by a pump [2]. In *digital* microfluidics, the droplets are moved on a planar-surface using electrowetting or dielectrophoresis [3], [4]. In this work, we focus on two-phase flow microfluidics. Here, the droplets flow through closed channels to modules executing standard unit operations like mixing, incubating, sorting, and sensing. For example, heating can be realized by a meander channel (which can automatically be designed using the "Meander Designer" proposed in [5]) under which a heating device is placed or merging and incubation can be realized by microfluidic traps as e.g. proposed by Garstecki's and Ren's group [6], [7]. Comprehensive reviews of droplet operations and an overview of experiments are available in [8]–[10].

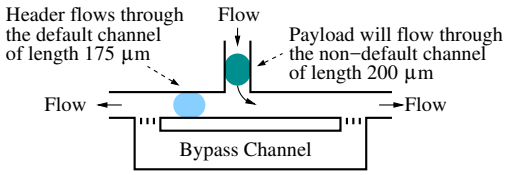


Fig. 1: Bifurcation

Despite the superior biocompatibility (e.g. the isolated droplets avoid cross contamination and washing steps [9]) as well as the simple and cheap fabrication of corresponding devices (usually realized with 3D printers, engraving machines, or soft-lithography processes using polymers as base material [11]–[13]), two-phase flow microfluidics bears the potential to realize multiple droplet paths. This yields *microfluidic networks* which allow multiple experiments on a single device [14].

To this end, a mechanism is required to route the droplets through the different paths of the microfluidic network. Instead of using valves, switches, or any other active components, this can be realized by exploiting hydrodynamic effects only – yielding a concept for *passive droplet routing*. This concept has been applied e.g. in *Networked Labs-on-Chips* [15] and *Hydrodynamic Controlled Microfluidic Networks* [16].

More precisely, a *bifurcation* as shown in Fig. 1 allows to route droplets through different paths. Therefore, the different volumetric flow rates (i.e. the amount of fluid which passes per time unit) in the successor channels of a bifurcation are exploited (in Fig. 1 these successor channels are named c_1 and c_2). These flow rates inversely depend on the fluidic resistances, which are mainly defined by the channel geometries (i.e. the smaller the diameter and/or the longer the channel, the higher the resistance) and viscosity of the continuous phase[1].

This droplet routing is based on the effect that a droplet always enters the channel with the highest volumetric flow rate [17], [18]. When the bifurcation does not contain any droplet, the larger amount of the flow enters the shorter successor channel (due to the lower resistance of this default channel). Hence, a single droplet will flow through the so-called *default successor* (cf. Fig. 1).

However, a droplet increases the overall resistance during its flow through a channel. In fact, droplets themselves increase the resistance of a channel e.g. through their viscosities,

[1] A *bypass* channel [17] connects the endpoints of the two successor channels. This bypass cannot be entered by any droplet and is used to make the droplet routing only dependent on the resistances of the successors.

droplet size, and geometry as studied e.g. in [14], [19], [20]. This has the effect that, during the flow of a droplet through the default successor, the larger amount of the flow now enters the non-default successor, i.e. the droplet *temporarily blocks* the default successor for following droplets. This allows to route a second, closely following, droplet into the *non-default successor* (cf. Fig. 1). Corresponding design guidelines, addressing schemes, validation through CFD-simulations, and initial experimental tests have been presented in [15], [16], [21]–[24]

These concepts of default successors at bifurcations and the possibility to route droplets with other droplets allow, in principle, to realize arbitrary paths through a microfluidic network. More precisely, if the actually considered droplet (containing the biological sample and called *payload* droplet in the following) is supposed to take a non-default successor at any bifurcation in the network, it has to made sure that another droplet (called *header* droplet in the following) arrives before and flows through the default successor. By this, the header temporarily blocks the default successor for closely following droplets. This is accordingly sketched in Fig. 1.

In order to route the payload along the desired path, it needs to be known *how many* headers are required and *when* to inject these relative to the payload. More precisely, the injected *droplet sequence* has to ensure that each bifurcation where a droplet is supposed to take the non-default successor is blocked by a header droplet. Furthermore, the time a header requires in order to flow into the channel to be blocked depends on the flow rates in the microfluidic network. These flow rates however constantly change due to the resistances caused by the flow of droplets. Furthermore, these flow interdependencies make it hard to estimate whether droplets unintentionally influence their respective ways or whether they merge. Thus far, no automatic solution has been proposed for this problem making it infeasible for larger networks to generate a proper droplet sequence executing the desired experiment.

In this work, we address this problem by providing an automatic method for the generation of droplet sequences realizing the desired experiment. The contributions of this paper can be summarized as follows:

- We propose an automatic method for generating droplet sequences on an abstract model. The therefore used abstract model was presented in [25] and allows to cope with the complex flow interdependencies.
- We validate the obtained droplet sequences using simulations. For the validation, we employed simulations as proposed in [26]–[28]. This validation guarantees that all interdependencies between droplets are considered and, hence, confirms the suitability of the obtained results.
- We evaluate the resulting automatic method confirming the capability to generate droplet sequences for large microfluidic networks.

The remainder of this paper is structured as follows: The next section first reviews the background on microfluidic networks including the flow distribution and how this distribution can be abstracted for design automation. Section III motivates the considered problem. Section IV describes the method for generating droplet sequences, which includes the determination of the droplet sequence on the abstract model and, afterwards, its validation by simulation. Results of our evaluation are summarized in Section V. Finally, the paper is concluded in Section VI.

II. Background

In this section, we review microfluidic networks, how the flow produced by the pumps distribute through these networks, and how the complex flow interdependencies can be abstracted in the form of a discrete model.

A. Microfluidic Networks and Experiments

In two-phase flow microfluidics, a pump produces a laminar flow of the continuous phase in which the droplets are injected. In order to support the passive droplet routing, the droplets need to be injected into the network at dedicated times. Therefore, corresponding droplet-on-demand components are utilized where a second pump produces a force on the dispersed phase and the droplets are generated with internal valves [29], external valves [30], or with pressure pulses [31]–[33].

A microfluidic network consists of a set of modules M executing unit operations and a set of channels C connecting these modules. In order to avoid that operations of modules are executed on headers, the modules are shielded by a droplet by size sorter [34]. A sorter steers payloads towards the module and forwards headers. Therefore, the sorter uses the different droplet sizes (i.e. droplet volumes) of headers and payloads. Finally, the network contains bifurcations allowing droplets to take multiple paths and, by this, to realize different experiments on a payload. Whether a path is implemented by the default- or by the non-default successor channel, is also defined by the network.

Example 1. *Consider the network shown in Fig. 2. Here, a pump produces a continuous flow in which payload and header droplets are injected using two droplet-on-demand components. The network consists of 19 channels* $C = \{c_1, \ldots, c_{19}\}$ *and four modules* $M = \{m, h, i, d\}$ *(for mixing, heating, incubating, and detecting). Each module is shielded by a sorter at its entry. Furthermore, this network contains two bifurcations where channel* c_4 *and channel* c_{11} *are the default successors because their lengths are shorter than* c_5 *and* c_{12}*, respectively. The resulting paths allow to implement e.g. the experiments* $(m,\ h,\ i,\ d)$*,* $(m,\ i,\ d)$*, as well as* $(m,\ h,\ d)$*.*

B. Flow Distribution

This section reviews the *one-dimensional (1D) analysis model* [35], [36], which allows to describe the flow distribution in microfluidic networks. In microfluidic devices, the flow usually occurs at low *Reynolds numbers* [12], [35] (the ratio of inertial to viscous forces) due to the small channel sections and relatively small flow rates. Hence, inertial effects such as gravity, separation, secondary flow, and turbulence are negligible. This allows to describe the flow using the *Hagen-Poiseuille's law* [37] with

$$\Delta P = Q \cdot R, \tag{1}$$

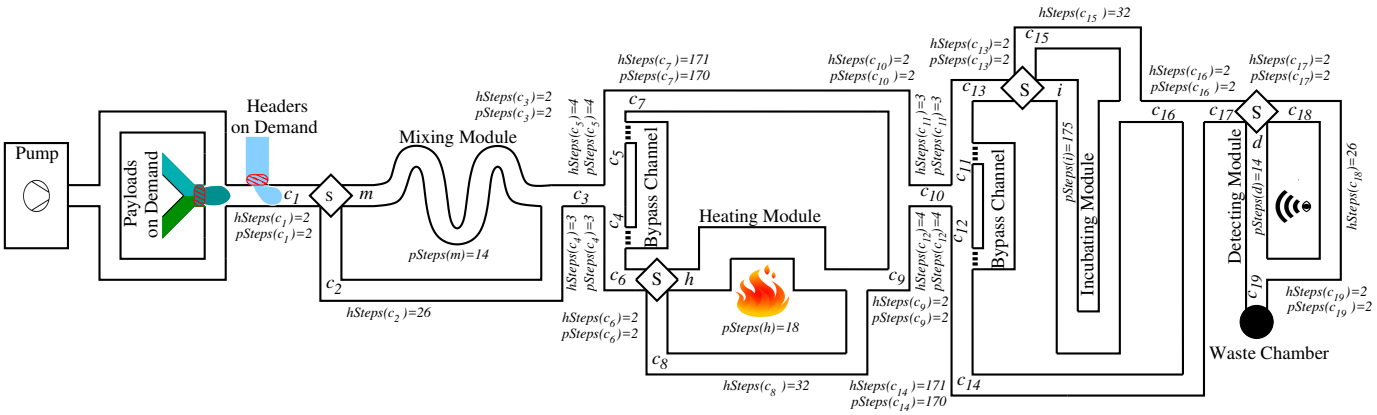


Fig. 2: Microfluidic network supporting passive droplet routing, which serves as a running example

where Q (in $[\mu l/min]$) is the *volumetric flow rate*, ΔP (in $[mbar]$) is the *pressure gradient*, and R (in $[mbar/(\mu l/min)]$) is the *fluidic resistance*.

This fluidic resistance of a channel/module is at a low Reynolds number constant and depends on the viscosity of the continuous phase and the geometry of the channel/module. For example, the resistance R_c of a rectangular channel c where the ratio of h_c/w_c is less than 1, is defined by

$$R_c = \frac{a\ \mu_{cont}\ l_c}{w_c\ h_c^3}, \tag{2}$$

where a denotes a dimensionless parameter defined as

$$a = 12\left[1 - \frac{192\ h_c}{\pi^5\ w_c}\tanh\left(\frac{\pi\ w_c}{2\ h_c}\right)\right]^{-1}. \tag{3}$$

The presence of droplets in channels/modules change the flow distribution as they cause additional resistances. This effect is used to route droplets through the system (cf. Fig. 1). The droplet resistance itself has been experimentally studied in several works such as e.g. [14], [19], [20].

In order to determine the flow distribution (i.e. the pressure gradients and the volumetric flow rates) in all channels and modules, the mass conservation and the relation described by the Hagen-Poiseuille equation can be applied [36]. This allows to obtain equations which are similar to the Kirchhoff's law, i.e. these laws can directly be transferred when we map the Hagen-Poiseuille equation to the *Ohm's law* with $V = R\ I$ (where the voltage V corresponds to the pressure gradient ΔP, the current I corresponds to the volumetric flow rate Q, and the resistance R of a conductor corresponds to the fluidic resistance R). More precisely, they can be described with the following rules:

- The sum of flow rates into a node is equal to the sum of flow rates out of that node. A node is a point in the network where a channel splits into multiple channels or where multiple channels merge to one channel.
- The directed sum of pressure gradients around any closed cycle is zero. The sign of the pressure gradients thereby depends on the direction of the flow rates.

The rules allow to obtain an equation system, which is used to determine the flow distribution in the channels and modules at a particular time and droplet state (i.e. their positions). That means the equation system has to be resolved every time a new droplet is injected or any droplet enters or exists a channel or module. By constantly re-computing the flow rates, they can be used to determine the droplets' velocities (i.e. by dividing the flow rate of a channel/module through its cross section) and durations required to pass channels/modules.

C. Discrete Model

As reviewed in the previous section, the droplets cause additional resistances and by their flow they constantly change the flow distribution. However, in order to cope with the complex flow interdependencies, we are employing the flow abstraction of microfluidic networks which has been introduced in [25] as a *discrete model* for design automation. This model allows to approximate the time a droplet requires to pass a channel/module as a discrete number of time steps. Since header and payload droplets have different volumes and, hence, cause different resistance changes, the discrete model approximates different numbers of time steps for them to pass channels/modules. More formally, the discrete model is defined as follows:

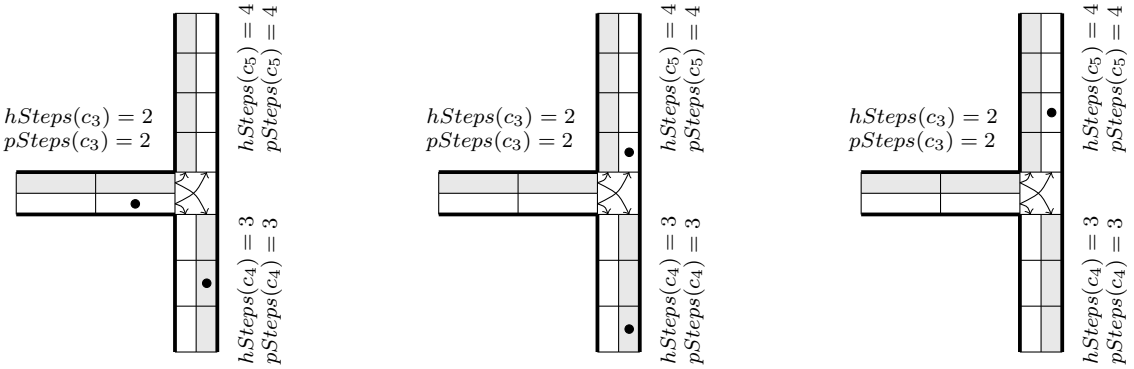
Definition 1. *The discrete model approximates the time a header/payload requires to pass a channel/module in an isolated manner, i.e. changes and interdependencies caused by other droplets are not considered. Then, the approximated time is discretized to a number of time steps by dividing it through the real time of one atomic time step given as T_a. This allows to define the number of time steps a payload requires to flow through a channel $c \in C$ or to execute a module $m \in M$ with the function*

$$pSteps : C \cup M \rightarrow \mathbb{N}. \tag{4}$$

Accordingly, the respective number of time steps for headers are defined by the function

$$hSteps : C \rightarrow \mathbb{N}. \tag{5}$$

Example 2. *Consider again the network shown in Fig. 2. The input for the discrete model is a full specification of the network (i.e. channel/module geometries, the input flow rate/pressure produced by the pump, and the viscosity of the*



(a) Time step t (b) Time step $t+1$ (c) Time step $t+2$

Fig. 3: The discrete model of [25] for a bifurcation with time steps for payloads (white segments) and headers (gray segments)

continuous phase) and of headers/payloads (i.e. viscosities and volumes). Using this full specification and assuming one time step to be equal to $T_a = 2\ ms$, the required number of time steps for all channels and modules can be derived, i.e. the functions $pSteps$ and $hSteps$ as annotated in Fig. 2 can be defined.

The discrete model also allows to predict the droplet routing as sketched in Fig. 1 and also the sorter element at the module entry. More formally, the discrete model predicts the path of a droplet as follows:

Definition 2. *The droplet behavior at a bifurcation (with successor channels of the same section) is determined by the number of time steps of the successors and other droplet positions. More precisely, by default a droplet flows into the successor requiring the least number of time steps (using the function of the considered droplet type). If this channel already contains a droplet, it flows into the other channel, i.e. non-default successor. This behavior is accordingly applied if both channels already contain a droplet. The discrete model also handles the sorter by steering the payload into the module and forwarding the headers.*

Example 3. *At the first bifurcation in Fig. 2 both, header and payload, require 3 time steps to flow through channel c_4 and 4 time steps to flow through channel c_5. Hence, c_4 is the default channel and c_5 is the non-default channel. This bifurcation including droplets at different time steps is visualized in Fig. 3. The segments in the channels represent the number of time steps a payload droplet (white background) or a header droplet (gray background) needs to pass this channel, i.e. they respectively represent the functions $pSteps$ and $hSteps$. Each of the Figs. 3a-3c represent the position of a payload and a header droplet at three consecutive time steps. As the default channel c_4 already contains a header droplet, the payload flows into the non-default channel c_5 in time step $t+1$.*

III. Considered Problem

The concepts of droplet routing allow to route a payload through different paths of a network. However, in order to establish a dedicated routing for the payload, payload and header droplets have to be injected into the network so that headers block the default successors which should not be taken by the payload. Therefore, the headers have to arrive right before the payload at corresponding bifurcations.

For ring networks as e.g. proposed in [16], [21], [22], [38], [39] the injection time of the header and payload droplets can be calculated by a formula because of the regular structure (i.e. the modules in a ring are connected in series and a re-injection mechanism of droplets closes this ring). However, this is not the case for more complex microfluidic networks. For example, for *application-specific* networks (which overcome severe shortcomings of ring networks) as proposed in [40], there is no formula possible which would allow to calculate the injection times. Here, the question is *how many* headers are needed and *when* to inject these headers relatively to the payload. In other words, what *droplet sequence* should be injected. In the following, this is discussed in more detail.

The path of the payload is predefined by the experiment to be executed. This path defines the bifurcations where the payload does flow along non-default successors and, hence, defines when the default successor has to be blocked by a header. In order to get a header blocking a default successor, multiple potential paths can be considered. Moreover, when a header takes a path which also contains non-default successors at bifurcations, additional headers are required, i.e. headers are required to route other headers.

Example 4. *Consider again the network shown in Fig. 2. In order to execute the experiment $(m,\ h,\ i,\ d)$, no header is required as the payload flows along the default successors at both bifurcations. Therefore, the simplest droplet sequence consisting of the payload only is sufficient.*

In order to execute the experiment $(m,\ i,\ d)$, one header is required to temporarily block the default successor c_4. This header can take only one possible path from c_1, c_2, c_3, into c_4. Therefore, the droplet sequence consists of a header and a payload.

Finally, in order to execute the third experiment $(m,\ h,\ d)$, the channel c_{11} needs to be blocked so that it is not taken by the payload. Therefore, a header can take two different paths, namely c_1, c_2, c_3, c_4, c_6, c_8, c_9, c_{10}, into c_{11} as well

as c_1, c_2, c_3, c_5, c_7, c_{10} into c_{11}. The second path requires an additional header which blocks the default successor c_4. In this case, a header is used to route another header and, overall, two headers are required.

In addition to selecting a possible set of headers and their paths, also their respective injection times are needed. Therefore, the time which a header requires in order to arrive at the bifurcation with the channel to be blocked has to be determined. This time depends on the flow rates in the network. However, these flow rates permanently change as the droplets cause additional resistances, i.e. whenever a new droplet is injected or any of the droplets in the network exits or enters another module/channel, all flow rates in the network change as reviewed in Section II-B. Furthermore, these complex interdependencies make it hard to determine whether droplets unintentionally influence their respective paths or whether droplets merge. Eventually, this may render originally intended paths impossible (as e.g. a header cannot timely arrive at the channel to be blocked) and, therefore, requires the consideration of alternatives. If no alternative is left, this is a clear indication that the considered network may not allow to execute that particular experiment at all [41].

These non-trivial interdependencies motivate an automatic method for the generation of droplet sequences since, in particular for larger networks, it is infeasible to conduct all corresponding considerations manually. Therefore, we propose a two-step approach for generating a droplet sequence: In the first step, a droplet sequence is generated on the discrete model described in Section II-C. This allows to determine the number of required headers as well as the (abstract) time steps when they are supposed to be injected. In the second step, the resulting droplet sequence is then validated through a simulation on the 1D analysis model. If the sequence is valid, i.e. indeed realizes the desired experiment, the process terminates. Otherwise, another droplet sequence is generated on the discrete model. In the following sections, both steps are described in detail.

IV. Generation of Droplet Sequences

In this section, we first introduce the used notation. Afterwards, the algorithmic details are described which consist of generating candidates of headers as well as their paths, determining respective injection times for a candidate, checking the resulting droplet sequence for consistency, and, finally, validating the obtained droplet sequence by simulation.

A. Used Notation

First, we introduce the notation which is used to describe the proposed droplet generation method. This includes a notation for paths, non-default successors, and the time a droplet needs to flow through the path.

Definition 3. *The path of the payload p is predefined by the experiment to be executed and can be described as a sequence over channels and modules, i.e. $path^p \in \{C \cup M\}^n$ with $1 \leq n \leq |C| + |M|$. A header h_k (with $k \in \mathbb{N}$ as its identifier) can take multiple paths through the network until it reaches the channel which should be blocked. This set of path options for h_k is described with $P^{h_k} = \{path_0^{h_k}, path_1^{h_k}, \ldots\}$ and each $path_i^{h_k}$ is described as a sequence over channels, i.e. $path_i^{h_k} \in C^n$ with $1 \leq n \leq |C|$.*

Whenever a payload or header path contains a non-default successor of a bifurcation, the corresponding default successor has to be blocked by a header. The non-default successors along a payload-path $path^p$ and a header-path $path_i^{h_k}$ are collected in the sets $nonDefault_{path^p} \subseteq C$ and $nonDefault_{path_i^{h_k}} \subseteq C$, respectively.

The required time which the payload p needs in order to flow from the injection point into a channel/module along its path $path^p$ is recursively defined by

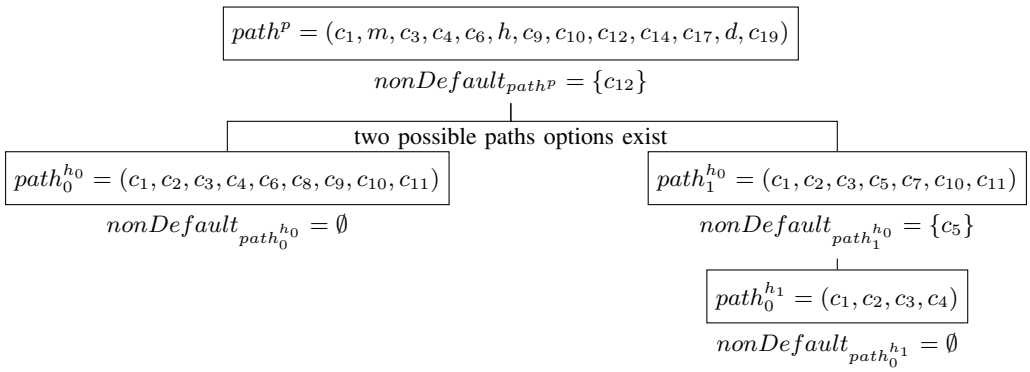
$$T_{path^p}(path^p[1]) = 0$$
$$T_{path^p}(path^p[j]) = T_{path^p}(path^p[j-1]) + pSteps(path^p[j-1]),$$

where the index notation $path^p[j]$ is used to access a channel/module at position j in the corresponding path. In a similar fashion, the required time for a header h_k to flow along path $path_i^{h_k}$ is defined by using the function $hSteps$ instead of $pSteps$.

B. Generating Candidates

Using the notation introduced above allows to generate possible sets of headers and their paths, so-called *candidates*, which can be used to realize a desired experiment. Therefore, the path of the payload $path^p$ serves as starting point. For this path, all non-default successors are collected in $nonDefault_{path^p}$. For each channel in $nonDefault_{path^p}$, a header h_k is required which blocks the corresponding default successor. To this end, all possible paths to each of the default successor are determined. These paths form possible options for the header h_k and are collected in P^{h_k}. Furthermore, these paths possibly contain non-default successor channels again. Therefore, in order to route these headers along the non-default successors, further headers are required. This procedure is recursively repeated for all determined paths until all $nonDefault$ sets are equal to $\emptyset$.

Example 5. *Consider the network shown in Fig. 2 and the experiment where the payload should flow through the modules (m, h, d). Assignments of the variables introduced before are shown in Fig. 4. First, the algorithm considers the path of the payload, i.e. $path^p$. The payload has to flow through the non-default successor c_{12} of the second bifurcation, i.e. $nonDefault_{path^p} = \{c_{12}\}$. Therefore, the default successor c_{11} needs to be blocked. This blocking is accomplished by using a header h_0. For this header, there are two path options: It can flow along $path_0^{h_0}$ and $path_1^{h_0}$ as defined in Fig. 4. The first option contains only default successors (i.e. $nonDefault_{path_0^{h_0}} = \emptyset$). The second option contains the non-default successor c_5. The blocking of the corresponding default successor c_4 can be done by a further header h_1 which has to flow along $path_0^{h_1}$ consisting of default successors only.*



- Paths of the 1^{st} candidate: $path^p$, $path_0^{h_0}$
- Paths of the 2^{nd} candidate: $path^p$, $path_1^{h_0}$, $path_0^{h_1}$

Fig. 4: Candidate tree

This recursive procedure of determining path options produces a *candidate tree*, which contains all possible sets of header paths. Recall that the options arise from the different paths through which the headers can flow in order to get to the default channel which should be blocked.

In order to realize the experiment, a set of paths, i.e. a *candidate*, has to be selected. Therefore, the candidate tree is traversed top-down. At each point where a header has multiple path options, exactly *one* is selected. The other path options and their subsequent headers are not required at this point and, therefore, are not part of the currently considered candidate. Overall, this candidate tree exhaustively provides all potential candidates, i.e. contains all combinations of the path options.

Example 6. *Let's continue Example 5. Overall, two possible candidates of header paths exist: The first candidate requires a single header h_0 which flows along $path_0^{h_0}$. The second candidate requires two headers h_0 and h_1, which flow along $path_1^{h_0}$ and $path_0^{h_1}$, respectively. Both candidates including the payload path are summarized in the bottom of Fig. 4.*

However, not all of these candidates indeed result in a valid execution of an experiment, since (1) droplets may merge or (2) droplets may mutually influence their respective paths. Since fewer headers cause fewer mutual interdependencies, we check the candidates in ascending order with respect to their number of required headers.

C. Determining the Injection Times

As next step, the injection times for a candidate are determined. Again, the droplets contained in the selected candidate are traversed starting with the payload path: First, the algorithm starts with the payload p and sets its injection time to $t = t_t$. This allows to determine the payload's entering time in each non-default successor $c \in nonDefault_{path^p}$, i.e. $T_{path^p}(c) + t_t$. At the time step when the payload should enter a non-default successor, a header has to block the corresponding default successor denoted here as d. That means, the header has to flow through the default successor d, which causes a higher flow rate into the non-default successor allowing the payload to enter this non-default successor.

For this header h_k, the earliest and latest time step at which the header can enter the default successor d can be determined by $T_{path^p}(c) + t_t - hSteps(d)$ and $T_{path^p}(c) + t_t - T_\Delta$, respectively. The earliest time step guarantees that the header is still in the default successor when the payload arrives at the bifurcation. The latest time step guarantees a minimum distance T_Δ between droplets. The entering time step of the header can be varied within this range. For example, if the header should be in the middle of the default successor when the payload arrives, it has to enter at $middle = T_{path^p}(c) + t_t - \lceil hSteps(d)/2 \rceil$.

Knowing when the header h_k should block the default successor and also knowing its path (i.e. $path_i^{h_k}$, which is defined by the selected candidate) allows to determine its injection time. More precisely, the time the header h_k requires until it flows into the default successor d is given by $T_{path_i^{h_k}}(d)$. This allows to determine the injection time of h_k by $Inj^{h_k} = middle - T_{path_i^{h_k}}(d)$.

Again, the injection times of the headers which are necessary to route h_k are recursively determined by repeating the steps from above. After the determination of all injection times, t_t is replaced with a number of time steps so that the earliest injection of any droplet starts at time step $t = 0$.

Example 7. *Let's continue Example 5 and 6 for which we determine injection time steps for the first candidate (i.e. header h_0 flowing along $path_0^{h_0}$). Therefore, the values of the functions $hSteps$ and $pSteps$ as provided in Fig. 2 are used. A timeline showing the time steps at which the payload p and the header h_0 are injected as well as at which they are supposed to enter the channels of the considered bifurcation is provided in Fig. 5.*

First, we assume that the payload p is injected at time step $t = t_t$. Then, the time step at which p arrives at the non-default successor c_{12} is $t = T_{path^p}(c_{12}) + t_t = t_t + 45$. Therefore, h_0 has to arrive in the default successor c_{11} before $t = t_t + 45$. Since h_0 stays 3 time steps in channel c_{11}, this means that h_0 should enter c_{11} at time step $45 + t_t - hSteps(c_{11}) = 45 + t_t - 3 = t_t + 42$ at the earliest and at time step $45 + t_t - 1 = t_t + 44$ at the latest (assuming

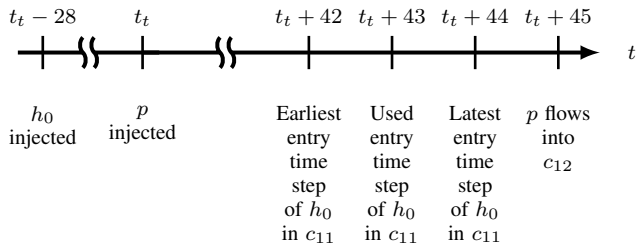


Fig. 5: Timeline

a minimum distance of $T_\Delta = 1$ time steps to the payload). Here, we want h_0 to be in the middle of c_{11} when p arrives, which gives $45 + t_t - \lceil 3/2 \rceil = t_t + 43$ as the entering time. As the header h_0 flows along $path_0^{h_0}$, it needs 71 time steps in order to arrive at c_{11}, i.e. $T_{path_0^{h_0}}(c_{11}) = 71$. Therefore, h_0 is injected at time step $Inj^{h_0} = 43 + t_t - 71 = t_t - 28$. For the considered candidate, no more injection times have to be determined. Hence, in the last step, we replace t_t with 28, so that the earliest injection starts at $t = 0$.

D. Checking for Consistency

Next, the generated droplet sequence is checked for consistency within the discrete model, e.g. it is checked whether no droplets influence their respective ways and whether the droplets meet a minimum distance T_Δ (note that the minimum distance T_Δ is an input parameter of the proposed method and, hence, can freely be chosen by the engineer). For example, when a header should flow into the default successor which is already occupied by another droplet, it flows into the non-default successor – resulting in a wrong path for the header. For this consistency check, the droplet positions at all time steps are checked.

However, even if a droplet sequence passes all these checks, it still might be possible that physical interdependencies prevent a successful execution of the experiment. Because of this, the resulting sequence has to be validated using simulation, which is described next.

E. Validation of Droplet Sequences

The discrete model is useful to efficiently generate droplet sequences. However, it abstracts the flow interdependencies between droplets. Therefore, in the second step of the proposed approach, a droplet sequence is validated through simulation.

To this end, simulation tools as proposed in [26]–[28] (and publicly available at http://iic.jku.at/eda/research/microfluidics_simulation/) can be used. These tools employ the concepts reviewed in Section II-B to simulate a generated droplet sequence and trace the path of the payload as well as all headers through the network – while, at the same time, considering all physical interdependencies which may affect the flow of the droplets. Therefore, if this simulation confirms that the payload is routed along the desired path, the sequence is valid and realizes the experiment.

The validation is conducted in a fully automated fashion: The discrete times of the droplet sequence are multiplied by the "real time" of an atomic time step T_a. The resulting droplet sequence and the network including all physical specifications (as e.g. the input flow rate/pressure gradient produced by the pump, the channel and module geometries, the viscosities of the continuous and dispersed phase, as well as the droplet sizes) are validated by simulations. This simulation allows to predict the path of the droplets. If the results show that the payload flows along the desired path, the sequence is considered valid.

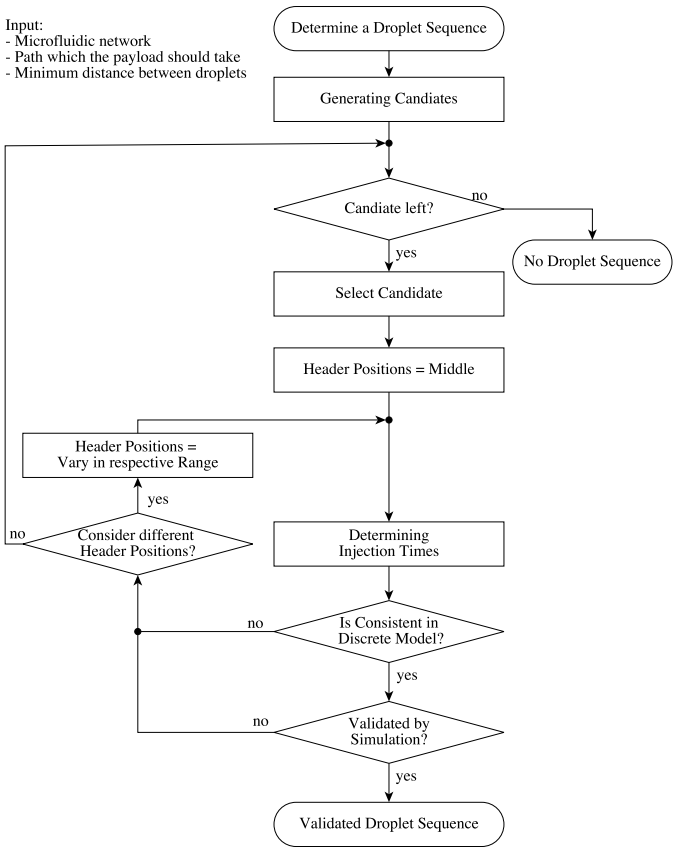


Fig. 6: Overview of the overall algorithm

F. Overall Method

Overall, the inputs and the steps executed by the proposed automatic method are summarized in Fig. 6. First, all possible sets of headers and their paths are generated in the form of the candidate tree (as described in Section IV-B). Then, the candidates are checked one after another whether they allow to correctly route the payload along the desired path. Therefore, the injection times of the headers and payloads for a candidate are determined (as described in Section IV-C). Finally, the resulting droplet sequence is checked for consistency on the discrete model (as described in Section IV-D) and validated using simulation (as described in Section IV-E). In case that either the consistency check or the validation fails, a new droplet sequence is generated by varying the position of the headers in the default channels (within the range as e.g. discussed in Example 7) or by using a different candidate.

By this algorithm, the possible droplet sequences are exhaustively checked until a valid one is found. If no alternative

TABLE I: Evaluation

Experiment	#Headers	#Checked Candidates	Valid?	Time [s]
Microfluidic Network with 8 Modules, 35 Channels, 3 Bifurcations				
Exp. 1	1	1	✓	< 1
Exp. 2	1	1	✓	< 1
Exp. 3	1	1	✓	< 1
Microfluidic Network with 10 Modules, 67 Channels, 8 Bifurcations				
Exp. 1	0	1	✓	< 1
Exp. 2	2	1	✓	< 1
Exp. 3	3	1	✓	< 1
Exp. 4	3	1	✓	< 1
Exp. 5	3	1	✓	< 1
Exp. 6	2	1	✓	< 1
Exp. 7	3	1	✓	< 1
Exp. 8	9	4	✓	273
Microfluidic Network with 12 Modules, 82 Channels, 10 Bifurcations				
Exp. 1	0	1	✓	< 1
Exp. 2	1	1	✓	< 1
Exp. 3	1	1	✓	< 1
Exp. 4	2	1	✓	< 1
Exp. 5	2	1	✓	< 1
Exp. 6	4	2	✓	3
Exp. 7	6	2	✓	17
Exp. 8	5	1	✓	< 1
Exp. 9	2	1	✓	< 1
Exp. 10	8	26	✓	12866
Microfluidic Network with 15 Modules, 101 Channels, 12 Bifurcations				
Exp. 1	2	1	✓	< 1
Exp. 2	4	2	✓	9
Exp. 3	5	12	✓	55
Exp. 4	2	1	✓	< 1
Exp. 5	2	1	✓	< 1
Exp. 6	3	1	✓	2
Exp. 7	6	4	✓	35
Exp. 8	2	1	✓	< 1
Exp. 9	3	1	✓	< 1
Exp. 10	5	1	✓	4
Exp. 11	5	1	✓	7
Exp. 12	7	1	✓	< 1
Microfluidic Network with 17 Modules, 118 Channels, 15 Bifurcations				
Exp. 1	1	1	✓	< 1
Exp. 2	8	7	✓	1259
Exp. 3	2	1	✓	< 1
Exp. 4	4	1	✓	< 1
Exp. 5	10	12	✓	1773
Exp. 6	6	3	✓	156
Exp. 7	2	1	✓	< 1
Exp. 8	4	1	✓	< 1
Exp. 9	4	1	✓	< 1
Exp. 10	6	2	✓	76
Exp. 11	12	1	✓	1398
Exp. 12	4	1	✓	4
Exp. 13	-	-	-	-
Exp. 14	12	2	✓	7479

is left, it is a clear indication that the considered network may not allow to execute that particular experiment at all.

V. Evaluation

The two steps of the proposed approach have been implemented in Java and Matlab resulting in an automatic method for generating droplet sequences. For the validation using simulation on the 1D analysis model, the simulator of [28] has been integrated into this automatic method. In order to evaluate the approach, we used five application-specific architectures generated with the method from [40] out of benchmarks from [42]. The resulting microfluidic networks were automatically dimensioned using the method from [43]. All evaluations have been conducted on a 4.2 GHz Intel Core i7 machine with 32GB of memory running 64-bit Ubuntu 16.04.

The obtained results are summarized in Table I. For all experiments per network, we provide the number of required headers in order to route the payload along the desired path (column "#Headers"), the number of checked candidates until a valid sequence was found (column "#Checked Candidates"), whether the obtained droplet sequence is valid in the 1D analysis model (column "Valid?"), as well as the required run-time in CPU-seconds to obtain that sequence (column "Time")[2].

The results show that all obtained droplet sequences are valid in the 1D analysis model. Furthermore, the desired droplet sequences can be realized in negligible run-times for the vast majority of experiments to be realized. But the results also show that the complexity increases with an increasing number of bifurcations. In fact, more bifurcations also result in more channels which might have to be blocked – increasing the number of headers (which in turn also need to be routed). This may even lead to situations where much more candidates have to be generated and validated as more droplets in the network also increase the probability of unintended blockings of channels, unintended merging, etc. Accordingly, the number of candidates and, hence, the run-time of the proposed method increases (which is not a limiting factor as the droplet sequences are generated only once upfront the experiment is executed). In the worst case, this may even lead to scenarios where no droplet sequence realizing the desired experiment at the given network can be generated at all. This is the case for one experiment of the last network, which gives a clear indication that the designer should revise this network, e.g. by removing bifurcations.

Using the method proposed in this work, droplet sequences can automatically be generated. Moreover, through the validation, it is guaranteed that all interdependencies between droplets are considered and, hence, the obtained results are indeed suitable.

VI. Conclusion

In this work, we presented the first automatic method for generating droplet sequences for microfluidic networks, which finally allow to route the payload droplet along a path executing the desired experiment. In order to handle the complex flow interdependencies in microfluidics, we proposed a two-step approach: First, a droplet sequence is generated on an abstraction, i.e. on a discrete model. Second, in order to prove that this droplet sequence correctly routes the payload along the desired path, the droplet sequence is validated through simulation. This two-step approach determines droplet sequences, which eventually allow to passively route droplets in microfluidic networks.

References

[1] B. Kintses, L. D. van Vliet, S. R. Devenish, and F. Hollfelder, "Microfluidic droplets: new integrated workflows for biological experiments," *Current Opinion in Chemical Biology*, vol. 14, no. 5, pp. 548–555, 2010.

[2]Note that the run-time includes the time needed to generate the desired droplet sequence as well as the simulation time needed to validate it.

[2] J. Atencia and D. J. Beebe, "Controlled microfluidic interfaces," *Nature*, vol. 437, no. 7059, p. 648, 2005.
[3] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab on a Chip*, vol. 2, no. 2, pp. 96–101, 2002.
[4] S. Haeberle and R. Zengerle, "Microfluidic platforms for Lab-on-a-Chip applications," *Lab on a Chip*, vol. 7, pp. 1094–1110, 2007.
[5] A. Grimmer, P. Frank, P. Ebner, S. Häfner, A. Richter, and R. Wille, "Meander designer: Automatically generating meander channel designs," *Micromachines – Journal of Micro/Nano Sciences, Devices and Applications*, vol. 9, no. 12, 2018.
[6] P. M. Korczyk, L. Derzsi, S. Jakieła, and P. Garstecki, "Microfluidic traps for hard-wired operations on droplets," *Lab on a Chip*, vol. 13, no. 20, pp. 4096–4102, 2013.
[7] X. Chen and C. L. Ren, "A microfluidic chip integrated with droplet generation, pairing, trapping, merging, mixing and releasing," *RSC Advances*, vol. 7, no. 27, pp. 16 738–16 750, 2017.
[8] A. Huebner, S. Sharma, M. Srisa-Art, F. Hollfelder, J. B. Edel *et al.*, "Microdroplets: a sea of applications?" *Lab on a Chip*, vol. 8, no. 8, pp. 1244–1254, 2008.
[9] S.-Y. Teh, R. Lin, L.-H. Hung, and A. P. Lee, "Droplet microfluidics," *Lab on a Chip*, vol. 8, pp. 198–220, 2008.
[10] S. Mashaghi, A. Abbaspourrad, D. A. Weitz, and A. M. van Oijen, "Droplet microfluidics: A tool for biology, chemistry and nanotechnology," *Trends in Analytical Chemistry*, vol. 82, pp. 118–125, 2016.
[11] D. C. Duffy, J. C. McDonald, O. J. Schueller, and G. M. Whitesides, "Rapid prototyping of microfluidic systems in poly (dimethylsiloxane)," *Analytical Chemistry*, vol. 70, no. 23, pp. 4974–4984, 1998.
[12] H. Gu, M. H. Duits, and F. Mugele, "Droplets formation and merging in two-phase flow microfluidics," *International Journal of Molecular Sciences*, vol. 12, no. 4, pp. 2572–2597, 2011.
[13] L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, A. Nicolosi, and M. Reno, "Experimental validation of a simple, low-cost, T-junction droplet generator fabricated through 3D printing," *Journal of Micromechanics and Microengineering*, vol. 25, no. 3, p. 035013, 2015.
[14] M. J. Fuerstman, A. Lai, M. E. Thurlow, S. S. Shevkoplyas, H. A. Stone, and G. M. Whitesides, "The pressure drop along rectangular microchannels containing bubbles," *Lab on a Chip*, vol. 7, no. 11, pp. 1479–1489, 2007.
[15] E. De Leo, L. Galluccio, A. Lombardo, and G. Morabito, "Networked labs-on-a-chip (NLoC): Introducing networking technologies in microfluidic systems," *Nano Communication Networks*, vol. 3, no. 4, pp. 217–228, 2012.
[16] E. De Leo, L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, and L. M. Zanoli, "Communications and switching in microfluidic systems: Pure hydrodynamic control for networking Labs-on-a-Chip," *Trans. on Communications*, vol. 61, no. 11, pp. 4663–4677, 2013.
[17] G. Cristobal, J.-P. Benoit, M. Joanicot, and A. Ajdari, "Microfluidic bypass for efficient passive regulation of droplet traffic at a junction," *Applied Physics Letters*, vol. 89, no. 3, pp. 34 104–34 104, 2006.
[18] T. Glawdel, C. Elbuken, and C. Ren, "Passive droplet trafficking at microfluidic junctions under geometric and flow asymmetries," *Lab on a Chip*, vol. 11, no. 22, pp. 3774–3784, 2011.
[19] T. Glawdel and C. L. Ren, "Global network design for robust operation of microfluidic droplet generators with pressure-driven flow," *Microfluidics and Nanofluidics*, vol. 13, no. 3, pp. 469–480, 2012.
[20] A. Biral and A. Zanella, "Introducing purely hydrodynamic networking functionalities into microfluidic systems," *Nano Communication Networks*, vol. 4, no. 4, pp. 205–215, 2013.
[21] E. D. Leo, L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, and L. M. Zanoli, "Design and assessment of a pure hydrodynamic microfluidic switch," in *Int'l Conf. on Communications*, 2013, pp. 3165–3169.
[22] L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "Microfluidic networks: Design and simulation of pure hydrodynamic switching and medium access control," *Nano Communication Networks*, vol. 4, no. 4, pp. 164–171, 2013.
[23] ——, "On the assessment of microfluidic switching capabilities in NLoC networks," in *Int'l Conf. on Nanoscale Computing and Communication*, 2014, p. 19.
[24] ——, "μ-NET: a network for molecular biology applications in microfluidic chips," *Trans. on Networking*, 2015.
[25] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "A discrete model for Networked Labs-on-Chips: Linking the physical world to design automation," in *Design Automation Conference*, 2017, pp. 50:1–50:6.
[26] A. Grimmer, M. Hamidović, W. Haselmayr, and R. Wille, "Advanced simulation of droplet microfluidics," *arXiv preprint arXiv:1810.01164*, 2018.
[27] A. Grimmer, X. Chen, M. Hamidović, W. Haselmayr, C. L. Ren, and R. Wille, "Simulation before fabrication: a case study on the utilization of simulators for the design of droplet microfluidic networks," *RSC Advances*, vol. 8, pp. 34 733–34 742, 2018. [Online]. Available: http://dx.doi.org/10.1039/C8RA05531A
[28] A. Biral, D. Zordan, and A. Zanella, "Modeling, simulation and experimentation of droplet-based microfluidic networks," *Trans. on Molecular, Biological, and Multi-scale Communications*, vol. 1, no. 2, pp. 122–134, 2015.
[29] R. Thakur, Y. Zhang, A. Amin, and S. Wereley, "Programmable microfluidic platform for spatiotemporal control over nanoliter droplets," *Microfluidics and Nanofluidics*, vol. 18, no. 5-6, pp. 1425–1431, 2015.
[30] K. Churski, M. Nowacki, P. M. Korczyk, and P. Garstecki, "Simple modular systems for generation of droplets on demand," *Lab on a Chip*, vol. 13, no. 18, pp. 3689–3697, 2013.
[31] S. A. Vanapalli, A. G. Banpurkar, D. van den Ende, M. H. Duits, and F. Mugele, "Hydrodynamic resistance of single confined moving drops in rectangular microchannels," *Lab on a Chip*, vol. 9, no. 7, pp. 982–990, 2009.
[32] A. J. Teo, K.-H. H. Li, N.-T. Nguyen, W. Guo, N. Heere, H.-D. Xi, C.-W. Tsao, W. Li, and S. H. Tan, "Negative pressure induced droplet generation in a microfluidic flow-focusing device," *Analytical Chemistry*, vol. 89, no. 8, pp. 4387–4391, 2017.
[33] M. Hamidović, W. Haselmayr, A. Grimmer, and R. Wille, "Towards droplet on demand for microfluidic networks," in *Workshop on Molecular Communications*, 2018, pp. 1–2.
[34] Y.-C. Tan, Y. L. Ho, and A. Lee, "Microfluidic sorting of droplets by size," *Microfluidics and Nanofluidics*, vol. 4, no. 4, pp. 343–348, 2008.
[35] K. W. Oh, K. Lee, B. Ahn, and E. P. Furlani, "Design of pressure-driven microfluidic networks using electric circuit analogy," *Lab on a Chip*, vol. 12, no. 3, pp. 515–545, 2012.
[36] M. Schindler and A. Ajdari, "Droplet traffic in microfluidic networks: A simple model for understanding and designing," *Physical Review Letters*, vol. 100, no. 4, p. 044501, 2008.
[37] H. Bruus, *Theoretical microfluidics*. Oxford university press Oxford, 2008, vol. 18.
[38] W. Haselmayr, A. Biral, A. Grimmer, A. Zanella, A. Springer, and R. Wille, "Addressing multiple nodes in Networked Labs-on-Chips without payload re-injection," in *Int'l Conf. on Communications*, 2017.
[39] G. Castorina, M. Reno, L. Galluccio, and A. Lombardo, "Microfluidic networking: Switching multidroplet frames to improve signaling overhead," *Nano Communication Networks*, vol. 14, pp. 48–59, 2017.
[40] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "Design of application-specific architectures for Networked Labs-on-Chips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 193–202, 2018.
[41] ——, "Verification of Networked Labs-on-Chip architectures," in *Design, Automation and Test in Europe*, 2017, pp. 1679–1684.
[42] M. F. Schmidt, "Microfluidic flow-based biochips," https://sites.google.com/site/mlsibiochips/, 2012.
[43] A. Grimmer, W. Haselmayr, and R. Wille, "Automated dimensioning of Networked Labs-on-Chip," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

Andreas Grimmer received the Master's degree in computer science from the Johannes Kepler University Linz, Austria, in 2015. From 2013 to 2015 he was a student researcher at the laboratory for Monitoring and Evolution of Very-Large-Scale Software Systems. Currently, he is a researcher at the Institute of Integrated Circuits and is working towards a PhD. His research interests include the design, verification, and test of circuits and systems with a particular focus on microfluidics.

Werner Haselmayr (S'08—M'13) received the Dipl.-Ing. degree in telematics from the Graz University of Technology, Austria and the Dr. techn. degree in mechatronics from the Johannes Kepler University Linz, Austria, in 2007 and 2013, respectively. He is currently an Assistant Professor at the Institute for Communications Engineering and RF-Systems at Johannes Kepler University. His research interests include algorithm design for wireless communications, iterative processing and molecular communication.

Rober Wille received the Diploma and Dr.-Ing. degrees in computer science from the University of Bremen, Germany, in 2006 and 2009, respectively. He has been with the Group of Computer Architecture, University of Bremen, Germany, from 2006––2015 and with the German Research Center for Artificial Intelligence (DFKI), Bremen, Germany, from 2013 onwards. Additionally, he worked as lecturer at the University of Applied Science of Bremen, Germany, and as Visiting Professor at the University of Potsdam, Germany, and the Technical University Dresden, Germany. Since 2015, he is Full Professor at the Johannes Kepler University Linz, Austria. His research interests are in the design of circuits and systems for both conventional and emerging technologies. He published more than 200 papers in journals and conferences and served as editor e.g. for TCAD as well as in program committees of numerous conferences such as ASP-DAC, DAC, DATE, and ICCAD.