Verification of Networked Labs-on-Chip Architectures

Andreas Grimmer*, Werner Haselmayr†, Andreas Springer†, Robert Wille*‡
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†Institute for Communications Engineering and RF-Systems, Johannes Kepler University Linz, Austria
‡Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{andreas.grimmer, werner.haselmayr, andreas.springer, robert.wille}@jku.at

***Abstract*—*Labs-on-Chips* (LoCs) revolutionize conventional biochemical processes and may even replace laboratories by integrating and minimizing their functionalities on a single chip. In a promising and emerging realization of LoCs, small volumes of reagents, so-called droplets, transport the biological sample and flow in closed channels of sub-millimeter diameters. This realization is called *Networked Labs-on-Chips* (NLoCs). The architecture of an NLoC defines different paths through which the droplets can flow. These paths are realized by splitting channels into multiple successor channels – so-called bifurcations. However, whether the architecture indeed allows to route droplets along the desired paths and, hence, correctly executes the intended experiment is not guaranteed. In this work, we present the first automatic solution for verifying whether an NLoC architecture allows to correctly route the droplets. Our evaluations demonstrate the applicability and importance of the proposed solution on a set of NLoC architectures.**

I. Introduction

Labs-on-Chips (LoCs) enable the miniaturization, integration and automation of chemical and biomedical procedures [1]. Their dissemination and exploitation significantly increased in the last decade [2], [3]. LoCs combine different laboratory functions on a single chip and are successfully used e.g. for in-vitro diagnostics, DNA sequencing, cell analysis, organism analysis, drug screening, or protein crystallization.

A possible platform are *droplet-based microfluidics* (see e.g. [4]), where tiny volumes (in the order of few micro- to pico-liters) of reagents – so-called *droplets* – are manipulated and controlled. In a promising, emerging realization, the droplets flow in *closed channels* of sub-millimeter diameters. These closed channels prevent evaporation and unwanted contamination and, hence, allow for a long-term incubation and storage of droplets [3], [5]. In this *channel-based* realization, an external pump generates a hydrodynamic force that drives the droplets through the system. Besides that, the system contains *modules*, which execute chemical/biological operations on the droplets. In order to enable the flow of droplets between modules, these modules are connected by channels. Overall, modules and channels represent the architecture through which the droplets flow – yielding a so-called *Networked Labs-on-Chip* (NLoC, [6], [7]).

NLoC architectures define multiple paths through which the droplets can flow. These paths are realized by *bifurcations* of channels, i.e. the splitting of a channel in two or more successor channels. When a droplet arrives at a bifurcation, it will flow along the successor channel with the lowest *hydraulic resistance* (mainly defined by the channel's geometry; see [7]–[10]). By that, the droplet itself increases the channel's hydraulic resistance and, therefore, temporarily "blocks" this channel for following droplets. This principle of selective blocking is eventually used to route a respective *payload* droplet (transporting the biological sample) to the desired modules to be executed. To this end, so-called *header* droplets are utilized which correspondingly block channels that must not be taken by the payload droplet.

Overall, this leads to complex NLoC architectures in which various experiments have to be realized. These experiments are realized by different paths through the NLoC architecture, which all need to be executable. However, for a given architecture it is not guaranteed that the payload droplet can indeed be routed along the path defining the required modules. At the same time, it is a non-trivial task to verify a given NLoC architecture (as will be illustrated in more detail later in Sec. III).

In this work, we present an automatic verification solution for NLoCs, which, for the first time, *verifies* whether all desired experiments can indeed be conducted on a given NLoC architecture. To this end, all possible droplet sequences and, hence, routings are symbolically considered. Afterwards, satisfiability solvers are applied to determine the droplet sequences which show the executability of the desired experiments and, by this, witness the correctness of the architecture. An evaluation demonstrates the applicability and importance of the proposed verification solution on a set of NLoC architectures. This allows to prove whether an NLoC architecture permits to execute all experiments.

II. Networked Labs-on-Chip

In *Networked LoCs* (NLoCs, [6], [7]), a *pump* produces a hydrodynamic force, which drives droplets through closed microchannels. In order to realize an experiment, a *payload* droplet has to traverse the desired sequence of *modules*, which includes elementary operations such as mixing, splitting, fusing, detecting, or heating. To enable the flow of droplets from one module to another, the modules are connected by *channels*.

In this section, we explain how the pump, the modules, and the channels can be employed to realize NLoC architectures, how the droplets can be routed, and how this allows to conduct experiments as well as finally define a discrete model for the droplet flow.

A. Architecture

An NLoC architecture consists of

- the *pump p*, which is responsible for the droplet generation and also produces the force driving the droplets through the architecture,
- a set of *channels C*, which allows for directed flows of droplets, and
- a set of *modules M*, which defines the available operations in an NLoC.

The architecture defines how the pump, the channels and the modules are connected. We denote both, the channels C and the modules M, as *entities*. Then, in the general case, an entity can have multiple predecessor entities and multiple successor entities. This is formally defined as follows:

978-3-9815370-8-6/17/$31.00

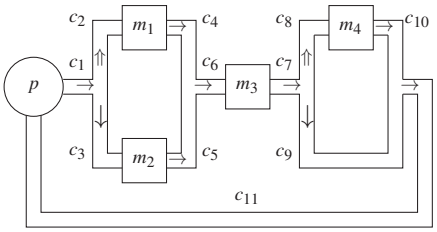

Fig. 1: NLoC architecture

Definition 1. *Let E be the* entities *of an NLoC architecture, i.e. the union of the channels C and modules M (i.e. $E := C \cup M$). Each entity $e \in E$ has a set of predecessor entities given by $pred : E \to \mathbb{P}(E)$ and a set of successor entities given by $succ : E \to \mathbb{P}(E)$. If e is the channel connected to the outlet of pump p, the function pred returns the empty set $\emptyset$. Accordingly, if e is the channel connected to the inlet of pump p, the function succ returns the empty set $\emptyset$.*

Example 1. *Consider the architecture shown in Fig. 1 comprised of a pump, four modules, and eleven channels. The pump p injects the droplets into the channel c_1. These droplets take one of the paths through the NLoC and return through channel c_{11} back to the pump p.*

The path which a droplet takes through the NLoC depends on the hydraulic resistances of channels and is described next.

B. Routing of Droplets

The droplet routing utilizes the different hydraulic resistances of channels. The hydraulic resistance of a channel is mainly defined by its *channel geometry*, i.e. the channel's diameter and channel's length [7], [11]. The smaller the diameter the higher the resistance and also the longer the channel the higher the resistance.

An architecture can define multiple paths through which the droplets can flow. The paths are realized by *bifurcations* of channels, i.e. the splitting of a channel in two or more successor channels. These successor channels have different hydraulic resistances. A droplet arriving at a bifurcation will flow along the channel with the lowest hydraulic resistance.

Example 2. *The architecture shown in Fig. 1 contains two bifurcations: channel c_1 has c_2 and c_3 as successor channels as well as channel c_7 has c_8 and c_9 as successor channels. Overall, these two bifurcations produce four paths, which all start from pump p and, finally, end in pump p.*

When the pump injects a single *droplet, it will flow in the first bifurcation along the channel c_2 because the hydraulic resistance of c_2 is lower than the hydraulic resistance of c_3 (denoted with a double arrow $\Uparrow$ in Fig. 1). Afterwards, this droplet will flow along channel c_8 because its hydraulic resistance is also lower than the hydraulic resistance of channel c_9. Finally, the droplet flows back to pump p.*

However, droplets themselves increase by their flow the hydraulic resistance of a channel [7], [9]. Therefore, droplets may change the hydraulic resistance of channels in a way that following droplets may take another path. In fact, droplets are used to increase the hydraulic resistances of channels and, hence, force the following droplets to take the desired channels. Overall, a droplet always flows into the channel with the lowest hydraulic resistance, if it does not already contain other droplets.

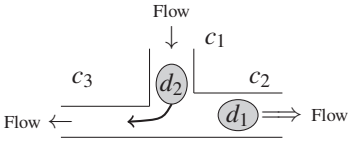

Fig. 2: A 2-way bifurcation of a channel

Example 3. *Fig. 2 shows the first 2-way bifurcation from the architecture of Fig. 1. The snap-shot shows two droplets d_1 and d_2. The droplet d_1 took the channel c_2 because, at the arrival time of d_1 at the bifurcation, the hydraulic resistance of c_2 was lower than those of c_3 (the diameter of c_2 is larger than the diameter of c_3). However, the following droplet d_2 will take the channel c_3, because d_1 increased the hydraulic resistance of c_2 so that the hydraulic resistance of c_3 is now lower than those of c_2.*

Videos at http://www.jku.at/iic/eda/nloc show physical realizations of the scenarios discussed above.

C. Realization of Experiments

An experiment is defined by a sequence of modules to be executed. To execute an experiment, the *payload* droplet has to be routed along the path of the architecture containing the desired sequence of modules. The experiment is finished when the payload droplet arrives at the pump[1].

Definition 2. *Let $\Phi := \{\phi_1, \phi_2, \ldots\}$ be the* set of experiments *to be conducted. Then, an* experiment $\phi \in \Phi$ *is a sequence of modules with $\phi \in M^n$, where $n \in \mathbb{N}$ is a natural number defining the number of modules in the experiment.*

The routing of the payload droplet is done by temporarily "blocking" those paths which the payload must not flow into. This blocking is achieved by using *header* droplets. The following example illustrates this routing.

Example 4. *Consider the experiments $\Phi := \{(m_1, m_3, m_4), (m_2, m_3, m_4), (m_2, m_3)\}$, which represent possible paths through the architecture given in Fig. 1.*

For example, in order to execute the experiment (m_1, m_3, m_4), the payload droplet has to traverse the specified modules. When the payload droplet arrives at the first bifurcation, it will flow into c_2 because it has a lower hydraulic resistance than c_3. After the payload droplet executed/passed m_1, c_4, c_6, m_3, and c_7, it arrives at the second bifurcation. Also here, the payload droplet flows into c_8, which is connected to m_4. Finally, the droplet flows through c_{10} and c_{11} back to the pump completing the experiment.

However, for executing the experiment (m_2, m_3, m_4), the payload droplet cannot just flow along the channels with the minimal hydraulic resistance. Instead a header *droplet has to be used to* route *the payload droplet. Therefore, a header droplet precedes the payload droplet, flows into channel c_2, and increases c_2's hydraulic resistance (i.e the header "blocks" this channel). By this, the following payload droplet correctly flows into channel c_3, which is connected to the desired module m_2. Afterwards, the module m_3 executes its operation on the payload droplet. Finally, the payload droplet takes the channel c_8 at the second bifurcation, which is connected to the last module m_4 of the experiment completing the experiment. The determination of the droplet sequence for the last experiment (m_2, m_3) is considered later in Sec. III.*

[1]Note that some pumps allow a re-injection of the same payload droplet and, therefore, allow to combine experiments.

As explained, in addition to the payload droplet, usually a number of header droplets is required to execute the experiments. These header droplets are injected by the pump, flow through the whole NLoC architecture and are eventually disposed in the pump. Furthermore, the modules have to forward these header droplets (i.e. the modules must not execute the operations on header droplets). Therefore, modules are equipped with *forwarding channels*. The required distinction[2] between the payload and header droplets is based on the different droplet sizes. Therefore, droplets must not coalesce (i.e. merge) with others because it would change their sizes and, hence, would break the distinction. Furthermore, a coalescence of the payload and a header droplet would destroy the biological sample.

D. Discrete Model

The time a droplet needs to pass/execute a channel or a module depends on the following constraints:

- The time a droplet needs for passing a *channel* mainly depends on its hydraulic resistance. But it also depends on all other channels and their composition in the NLoC.
- The time a droplet needs for passing a *module* depends, for a payload droplet, on the execution time of the operation and, for header droplets, on the time required to be forwarded to the outlet of the module.

The used *discrete* model simplifies the physical reality and is motived by the work of [10], which uses a *continuous* time and, hence, also continuous droplet positions within entities. In our model, we discretize the continuous time into time steps and do not consider all interrelations of droplets. This allows us to specify the time a droplet requires to pass/execute an entity as an amount of time steps. Therefore, we formally specify the time which droplets require to pass/execute a channel or a module as follows:

Definition 3. *The number of time steps a payload droplet requires to pass/execute an entity $e \in E$ is defined by the function $pSteps : E \to \mathbb{N}$. Accordingly, the function $hSteps : E \to \mathbb{N}$ defines the number of time steps a header droplet needs to pass an entity $e \in E$. Note that, in the following, we assume that payload and header droplets require an equal number of time steps to pass a channel, but the required time can differ for modules, i.e. depending on the droplet either the module executes an operation or it forwards the droplet.*

When defining the time step interval, it has to be ensured that the resulting distance between two "time step-consecutive" droplets prohibits an unexpected coalescence of these droplets. Alternatively, a minimum distance (specified as an amount of time steps) between droplets can be use to avoid unintended coalescences.

III. Problem Statement

Example 4 discussed two droplet sequences realizing the experiments (m_1, m_3, m_4) and (m_2, m_3, m_4) on the architecture given in Fig. 1. However, an NLoC architecture is only valid, if it allows to execute *all* experiments defined in Φ. Therefore, in this section we discuss the determination of droplet sequences intended to realize the last experiment (m_2, m_3) by using the discrete model introduced above.

Tab. I shows two different droplet sequences. Each column represents an entity, where the column widths define the required time steps a droplet needs to pass/execute the entity. Each row represents the position of payload and header droplets for a particular time step t. More precisely, whether a module or parts of a channel contain a header droplet, contain a payload droplet, or does not contain a droplet at all, is represented by an empty circle (○), a filled circle (●), or an empty cell ( ) respectively. As already discussed, sorters in modules route header droplets directly to the outlet of the module and this forwarding is assumed to require a single time step in this example. These forwarding channels are denoted with the suffix *FW* in the headers of Tab. I.

TABLE I: Droplet sequence attempts

First attempt:

t	c_1	c_2	m_1 FW	m_1	c_4	c_3	m_2 FW	m_2	c_5	c_6
1	○									
2	●	○								
3		●	○			●				
4				○		●				
5					○		●			
6					○			●		
7					○				●	
8									↯	

Second attempt:

t	c_1	c_2	m_1 FW	m_1	c_4	c_3	m_2 FW	m_2	c_5	c_6	m_3 FW	m_3	c_7	c_8	c_9	m_4 FW	m_4
1	○																
2		○															
3	●		○														
4				○		●											
5					○		●										
6					○			●									
7					○				●								
8										○	●						
9										●		○					
10											●		○				
11												●		○			
12													●				○
13														↯			

In the first droplet sequence (at the top of Tab. I), a header (denoted as ○) is injected in time step $t = 1$. At the first bifurcation, this header takes the channel c_2 because c_2's hydraulic resistance is lower than those of c_3 (i.e. Tab. I shows that c_2 requires 2 time steps compared to 3 time steps of c_3). In time step $t = 2$, the payload (denoted as ●) gets injected. Due to the fact that the header is still in channel c_2 (and, therefore, the hydraulic resistance of this channel is increased), the payload intentionally flows into channel c_3. After the payload passed channel c_3, the operation of module m_2 is executed in time step $t = 6$. However, in the chosen droplet sequence, the two droplets would coalesce afterwards in channel c_6 in time step $t = 8$. Therefore, the chosen sequence is invalid.

In the second droplet sequence (at the bottom of Tab. I), the header is injected in time step $t = 1$ and the payload is injected in time step $t = 3$. Therefore, it is avoided that the droplets would again coalesce in time step $t = 8$ as in the first sequence. When further considering the droplet's flow, we can determine that the header flows into channel c_8 in time step $t = 11$ as its hydraulic resistance is smaller than those of c_9. When the payload arrives at the second bifurcation in time step $t = 12$, the header is already in the forwarding channel of module m_4. Hence, the hydraulic resistance of channel c_8 is, again, less than those of channel c_9. This causes the payload to flow into channel c_8 and, finally, into module m_4. However, the requested experiment (m_2, m_3) does not contain module m_4. Therefore, this droplet sequence is invalid as well.

[2]This distinction between the payload and header droplets is done using an integrated sorter. For a possible passive realization see e.g. [12].

As illustrated by these two sequences, determining a droplet sequence, which correctly executes the given experiment (m_2, m_3), is a non-trivial task. This motivates the question whether there exists a valid droplet sequence for this experiment. Actually, for the given architecture no droplet sequence exists, which would realize the experiment (m_2, m_3). This motivates the following problem statement:

> ***Given*** *is the architecture including the time step specification of the entities E as well as the experiments* Φ.
> ***Wanted*** *is a verification whether all experiments* Φ *can be realized on the given architecture, i.e. whether for each experiment* $\phi \in \Phi$ *a droplet sequence can be determined, which routes the payload droplet through all modules defined in* ϕ.

IV. Verifying the Architecture

In this section, the proposed automatic solution to the problem motivated above is described. Therefore, we explain our verification strategy and its symbolic formulation.

A. Verification Strategy

In order to verify whether an architecture allows to execute an experiment, all possible droplet sequences have to be considered. To this end, it is important to know the maximum length of the droplet sequences to be considered (as this defines the search space). In fact, the given architecture bounds the maximum length of a droplet sequence to a finite number as follows:

The time a payload needs to execute an experiment is specified by the required modules and channels it flows through – denoted as t_ϕ. Only header droplets, which during the execution of an experiment are in the NLoC, can have an impact on the payload's path (i.e. on the taken channels). Therefore, we derive how many time steps headers can be injected *before* or *after* the payload and still may have an impact on the payload's path. Then, the time span between the *earliest* and the *latest* possible time defines the maximal length of the droplet sequence. This earliest and latest possible time is given as follows:

- The *earliest* possible time for injecting headers is given by the longest path of the architecture. This longest path is defined as the entities with the largest sum of time steps – denoted as $t_{maxPath}$ in the following. That means, if headers are injected at most $t_{maxPath}$ time steps *before* the payload, they can still be in the NLoC when the payload is injected. So these headers may have an impact on the payload's path.
- The *latest* possible time for injecting headers is given by the time required to execute the experiment (i.e. t_ϕ). Even if headers enter the NLoC *after* the payload, they potentially may impact the payload's path (i.e. headers may "overtake" the payload by taking another path and then block a channel). Therefore, the *latest* possible time a header can be injected and still may have an impact on the payload's path, is given by the time required to execute the experiment t_ϕ.

Definition 4. *Let T be the maximal length (i.e. the upper bound) of a droplet sequence, which is given by* $T = t_{maxPath} + t_\phi$.

This upper bound of the droplet sequence length guarantees that, if the architecture allows to execute an experiment ϕ, a valid droplet sequence is definitely within this bound. Otherwise, if within this upper bound no droplet sequence can be determined, we have verified that the given architecture does not allow to execute experiment ϕ.

However, even with this upper bound a significant amount of sequences have to be considered. More precisely, for an upper bound T, there are $T-1$ possible time steps, where either a header droplet can be injected or not – resulting in 2^{T-1} possibilities. Even if many of those can easily be excluded, still a huge (exponential) search space results.

Hence, we address this problem not by enumerating all possible droplet sequences and validating whether they execute the given experiment. Instead, we transform the determination of a droplet sequence as a decision problem, i.e. "Is there at least one valid droplet sequence for each experiment $\phi \in \Phi$, which correctly executes ϕ using a droplet sequence consisting of at most T time steps?". We symbolically formulate this decision problem as a *SAT Module Theories* (SMT, [13]) instance, which is passed to a corresponding solving engine[3].

B. Symbolic Formulation

This section provides the details of the symbolic formulation, which initially represents all (also invalid) droplet sequences and flows. Afterwards, we introduce constraints which ensure a valid droplet flow, enforce the execution of the experiment, and prevent the coalescence of droplets.

a) Symbolic Formulation of the Droplet Sequence and Droplet Positions: The droplet sequence is represented by two bit vectors. One bit vector represents the injection of the payload droplet and the other represents the injection of header droplets. Both vectors are of length T. More formally:

Definition 5. *Let $injP$ be the vector of length T representing the injection of the payload droplet. Further let $injH$ be the vector of length T representing the injection of the header droplets. The left-most bit represents $t = 1$ in our notation. Therefore, a 1 at position t in these bit vectors (i.e $injP[t] = 1$ or $injH[t] = 1$) represents the injection of a payload/header droplet in time step t. The time step in which the payload is injected is given by the upper bound, i.e. the payload is injected in time step $t_{maxPath}$. Therefore, the vector $injP$ contains a 1 at position $t_{maxPath}$ (i.e. $injP[t_{maxPath}] = 1$).*

In addition to the droplet sequences, a formulation is needed which symbolically represents the state (i.e. the droplet positions) of an NLoC at every time step. From the discussion above, we already know that there cannot be a droplet sequence longer than T time steps. Additionally, considering the maximal time steps that the *latest* possible injected droplet needs to flow back to the pump (given by the longest path), $T + t_{maxPath}$ time steps have to be considered. In all those time steps droplets must not coalesce.

Therefore, for all modules and channels and each time step $0 \leq t \leq T + t_{maxPath}$, two bit vectors representing the header droplets and the payload droplet are introduced, respectively. The lengths of the payload (header) bit vectors are defined by the function $pStep$ ($hSteps$) – cf. Def. 3. A symbolic one-hot encoding is applied to these bit vectors, where a 1 represents the presence of a payload/header droplet within a channel or module. When a payload (header) droplet enters a channel or module, the left most bit of the payload (header) bit vector is assigned 1. Then, the positions of the 1 in the vector represent how long droplets are already in the entity. More formally:

[3]In the past, corresponding SMT solvers have already successfully been applied in the design of electrowetting-based LoCs (see e.g. [14]) as well as continuous-flow-based LoCs (see e.g. [15]).

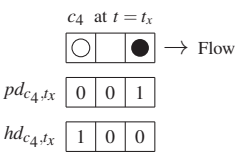

Fig. 3: Symbolic formulation of droplet positions

Definition 6. *For each entity $e \in E$ and each discrete time step $0 \leq t \leq T + t_{maxPath}$, we introduce a bit vector for the payload droplet $pd_{e,t}$ and for the header droplets $hd_{e,t}$. The lengths of the bit vectors $pd_{e,t}$ are defined by $pSteps(e)$ and the lengths of the bit vectors $hd_{e,t}$ are defined by $hSteps(e)$.*

Example 5. *Fig. 3 shows the channel c_4 in time $t = t_x$ with its currently contained droplets. Furthermore, it shows the corresponding bit vectors hd_{c_4,t_x} and pd_{c_4,t_x}. Their assignments represent the state of this channel c_4.*

Passing this symbolic formulation to a solving engine would yield an arbitrary assignment of the bit vectors and, hence, it is not guaranteed that the droplets correctly flow through the NLoC, that an experiment is executed, or that no droplets coalesce. Therefore, we have to restrict the assignments of the symbolic formulation.

b) Enforcing the Droplet Flow: The hydrodynamic force produced by the pump p causes a flow of the payload and header droplets through the NLoC. To realize this flow, the time a droplet is already contained in an entity has to be increased in each time step. The used one-hot encoding allows to do this with a single right shift. Note that this also handles a droplet leaving an entity because the execution of the right shift operation drops a 1 contained in the last position (i.e. the execution of the shift operation drops $hd_{e,t}[hSteps(e)]$ and $pd_{e,t}[pSteps(e)]$). In the following, we formally discuss the flow of the header droplets. Likewise, it is applied to the bit vectors representing payload droplets.

The flow of header droplets is formally enforced as

$$\bigwedge_{e \in E} \bigwedge_{t=1}^{T+t_{maxPath}} hd_{e,t} = (hd_{e,t-1} \gg 1) \vee \text{␣}. \tag{1}$$

Eq. 1 executes a single right shift on the bit vector $hd_{e,t-1}$ of time $t-1$ and assigns this value to bit vector $hd_{e,t}$. However, it does not consider the droplet flow between entities (e.g. a droplet leaving a module and entering the successor channel). Depending on the entity e, three cases have to be differentiated for the droplet flow between entities (replacing the place holder ␣ of Eq. 1):

- In case e is the channel connected to the outlet of pump p (i.e. $pred(e) = \emptyset$), the pump injects new droplets into this channel within the first T time steps. The injection sequence of headers is represented by the vector $injH$. A 1 (i.e. $injH[t] = 1$) represents the injection of a new header in time t. This motivates the following replacement:

$$\text{␣} = injH[t]. \tag{2}$$

- In case e is a channel after a bifurcation, a droplet only flows into channel e, if it has the lowest hydraulic resistance compared to the other successor channels of the bifurcation and it does not already contain droplets [7], [9]. Therefore, let f be the channel before the bifurcation (i.e. the predecessor of e). Three requirements have to be fulfilled so that a droplet flows into the channel e in time step t: (1) there has to be a header droplet, which leaves the predecessor f in the next time step t (i.e. part 1 of Eq. 3), (2) the channel e itself does not already contain droplets (which would have increased the hydraulic resistance) (i.e. part 2 of Eq. 3), and (3) all channels with a lower hydraulic resistance already contain other droplets (i.e. part 3 of Eq. 3). This motivates the following replacement:

$$\text{␣} = \underbrace{hd_{f,t-1}[hSteps(f)]}_{1} \wedge \underbrace{hd_{e,t-1} = 0 \wedge pd_{e,t-1} = 0}_{2} \wedge \underbrace{\bigwedge_{\{g \in succ(f) : hSteps(g) < hSteps(e)\}} hd_{g,t-1} > 0 \vee pd_{g,t-1} > 0}_{3}. \tag{3}$$

- In all other cases, it is necessary to check if one of the predecessors contains a header, which leafs the predecessor in the next time step t (i.e. $hd_{f,t-1}[hSteps(f)]$). If so, this header enters e. This motivates the following replacement:

$$\text{␣} = \bigvee_{f \in pred(e)} hd_{f,t-1}[hSteps(f)]. \tag{4}$$

These constraints are adopted to also ensure the flow of the payload droplet but, due to their similarity, are not explicitly shown here.

Passing this formulation to a solving engine now already ensures the correct flow of droplets. However, it does not ensure that the experiments are executed or that no droplets coalesce.

c) Enforcing the Experiment: Initially, before the execution of the experiment starts (at $t = 0$), we ensure that the NLoC does not already contain droplets, i.e.

$$\bigwedge_{e \in E} hd_{e,0} = 0 \wedge pd_{e,0} = 0. \tag{5}$$

The payload droplet has to be routed through the sequence of the desired modules defined by the experiment ϕ. In order to ensure this execution of ϕ, we enforce that each module defined in ϕ executes at some time step its operation on the payload droplet. Therefore, we enforce all bit vectors $pd_{e,t}$ representing the modules contained in ϕ to be greater than 1 at some time $1 \leq t \leq T$. In contrast, the payload droplet is not allowed to traverse any module not contained in the experiment (i.e. $M \setminus \phi$). Therefore we enforce those $pd_{e,t}$ vectors to be equal to 0 all time. This motivates the constraint:

$$\bigwedge_{e \in \phi} \bigvee_{t=1}^{T} pd_{e,t} > 0 \wedge \bigwedge_{e \in M \setminus \phi} \bigwedge_{t=1}^{T+t_{maxPath}} pd_{e,t} = 0. \tag{6}$$

d) Preventing Droplet Coalescence: Finally, it is only left to enforce that droplets never coalesce. Therefore, we have to ensure that always at most one droplet enters an entity in each time step t, i.e.

$$\bigwedge_{e \in E} \bigwedge_{t=1}^{T+t_{maxPath}} \Big(\sum_{f \in pred(e)} pd_{f,t}[pSteps(f)] \vee hd_{f,t}[hSteps(f)] \Big) \leq 1 \tag{7}$$

Passing the resulting formulation to a solving engine now only yields assignments representing valid droplet sequences and valid droplet flows, which correctly execute the experiment ϕ. If the solving engine determines that no assignment is possible satisfying all constraints, it has been proven that the given architecture does not allow to execute the experiment ϕ. If instead a satisfying assignment is determined, the assignments of the vectors $injP$ and $injH$ represent the droplet sequence executing the experiment ϕ. This process is repeated for all experiments $\phi \in \Phi$ and if there is a satisfying assignment for all experiments, the proposed solution has proven that the given architecture is capable of executing all experiments.

TABLE II: Evaluation results

NLoC	$\|M\|$	$\|C\|$	$\|\Phi\|$	Avg. ϕ Length	Max. T	Valid	Time [s]
NLoC architectures created by graphs from [16]:							
DAGmar8	8	21	5	4.0	84	✓	3
DAGmar10	10	28	10	7.0	151	✓	31
DAGmar12	12	28	8	7.1	182	✓	54
DAGmar14	14	35	14	7.4	181	✓	109
DAGmar16	16	31	9	9.7	239	✓	72
DAGmar18	18	41	24	12.0	261	✓	908
DAGmar20	20	44	36	14.0	309	✓	868
DAGmar22	22	49	96	12.5	292	✓	8905
DAGmar24	24	55	64	14.1	325	✓	8392
DAGmar8	8	21	6	4.0	84	✗	2
DAGmar10	10	28	11	7.0	151	✗	11
DAGmar12	12	28	9	7.3	184	✗	31
DAGmar14	14	35	15	7.5	182	✗	22
DAGmar16	16	31	10	9.8	241	✗	32
DAGmar18	18	41	25	12.0	261	✗	112
DAGmar20	20	44	37	14.0	309	✗	199
DAGmar22	22	49	97	12.5	292	✗	1026
DAGmar24	24	55	65	14.1	325	✗	2855
NLoCs architectures inspired and created from [17]:							
B1	8	15	3	6.3	137	✓	4
B2	10	30	8	6.0	137	✓	35
B3	12	36	10	7.7	222	✓	280
B4	15	46	12	7.8	201	✓	464
B5	17	53	14	9.2	260	✗	953

$|M|$: number of modules $|C|$: number of channels $|\Phi|$: number of experiments
Avg. ϕ Length: average length of the experiments Max. T: upper bound of time steps for all experiments Valid: validity of the architecture Time [s]: required run-time in CPU seconds

V. Evaluation

The proposed solution has been implemented in Java resulting in an automatic verification method for NLoC architectures. To this end, the SMT solver Z3 [18] in its latest version has been utilized. In order to evaluate the proposed solution, we used NLoC architectures generated using *DAGmar* [16] as well as NLoC architectures inspired and created from [17]. We use the proposed solution to efficiently verify whether these architectures allow to execute the given set of experiments. All experiments have been conducted on a 3.8 GHz Intel Core i7 machine with 32GB of memory running 64-bit Ubuntu 16.04.

The results are summarized in Tab. II. The first five columns provide the name, the number of modules ($|M|$), the number of channels ($|C|$) of the architecture followed by the considered number of experiments ($|\Phi|$), and the average length of these experiments (Avg. ϕ Length). Afterwards, the respective results are reported. First, the maximum of all droplet sequence upper bounds (Max. T) is provided. Second, we state whether an architecture does allow (denoted by ✓) or does not allow (denoted by ✗) to execute all experiments (Valid). Finally, the last column provides the required run-time (in CPU seconds).

The first part of Tab. II shows architectures generated using the graph library *DAGmar* [16]. These architectures can be scaled with respect to the number of modules, channels, and experiments. In this way, we were able to evaluate architectures of different size. To test both cases, i.e. executable experiments as well as nonexecutable experiments, we define two sets of experiments. The first set contains all executable experiments on a given architecture and the second set additionally contains a nonexecutable experiment. We group the first set in the upper half and the second set in the lower half of the first part of Tab. II.

For all architectures, the proof that no valid droplet sequence exists for one of the experiments takes less runtime than the proof that there are valid droplet sequences for all experiments. That is because as soon as one experiment cannot be executed, the verification can be aborted and the given architecture is proven invalid.

The NLoC architectures inspired and created from [17] are shown in the second part of Tab. II. Our verification solution shows that only four out of five NLoC architectures indeed allow to execute all experiments. For the last architecture (B5), which also represents the largest NLoC architecture with the most experiments, our solution verifies that not all experiments can correctly be executed. Hence, this architecture cannot be used to conduct all desired experiments. Overall, the results show that the proposed solution has acceptable run times for the considered NLoC architectures.

VI. Conclusions

In this work, we presented the first automated solution for verifying whether a Networked Labs-on-Chip architecture allows to execute all experiments. Therefore, the presented solution proves the existence or non-existence of a droplet sequence realizing an experiment on the given architecture. In order to efficiently tackle this verification problem, a symbolic formulation has been proposed. Afterwards, satisfiability solvers are utilized. Evaluations showed that the solution efficiently verifies NLoC architectures and confirmed the applicability and importance of a verification solution.

References

[1] A. J. Demello, "Control and detection of chemical reactions in microfluidic systems.," *Nature*, vol. 442, no. 7101, pp. 394–402, 2006.

[2] S. Haeberle and R. Zengerle, "Microfluidic platforms for Lab-on-a-Chip applications," *Journal on Lab on a Chip*, vol. 7, pp. 1094–1110, 2007.

[3] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, "Microfluidic Lab-on-a-Chip platforms: requirements, characteristics and applications," *Journal of Chemical Society Reviews*, vol. 39, no. 3, pp. 1153–1182, 2010.

[4] S.-Y. Teh, R. Lin, L.-H. Hung, and A. P. Lee, "Droplet microfluidics," *Journal on Lab on a Chip*, vol. 8, pp. 198–220, 2008.

[5] L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "μ-NET: a network for molecular biology applications in microfluidic chips," *IEEE/ACM Trans. on Networking*, 2015.

[6] E. De Leo, L. Galluccio, A. Lombardo, and G. Morabito, "Networked labs-on-a-chip (NLoC): Introducing networking technologies in microfluidic systems," *Journal of Nano Communication Networks*, vol. 3, no. 4, pp. 217–228, 2012.

[7] E. D. 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," *IEEE Trans. on Communications*, vol. 61, no. 11, pp. 4663–4677, 2013.

[8] 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*, pp. 3165–3169, 2013.

[9] L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "On the assessment of microfluidic switching capabilities in NLoC networks," in *Int'l Conf. on Nanoscale Computing and Communication*, p. 19, 2014.

[10] A. Biral, D. Zordan, and A. Zanella, "Modeling, simulation and experimentation of droplet-based microfluidic networks," *IEEE Trans. on Molecular, Biological, and Multi-scale Communications*, vol. 1, no. 2, pp. 122–134, 2015.

[11] E. D. Leo, L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "Microfluidic networks: design and test of a pure hydrodynamic switching function," in *Int'l Conf. on Communications*, pp. 787–791, 2013.

[12] Y.-C. Tan, Y. L. Ho, and A. P. Lee, "Microfluidic sorting of droplets by size," *Journal of Microfluidics and Nanofluidics*, vol. 4, no. 4, pp. 343–348, 2008.

[13] A. Biere, M. Heule, and H. van Maaren, *Handbook of satisfiability*, vol. 185. IOS Press, 2009.

[14] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, "Exact one-pass synthesis of digital microfluidic biochips," in *Design Automation Conference*, 2014.

[15] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Asia and South Pacific Design Automation Conference*, 2017.

[16] C. Bachmaier, A. Gleißner, and A. Hofmeier, "DAGmar: Library for DAGs," 2012.

[17] M. F. Schmidt, "Microfluidic flow-based biochips." https://sites.google.com/site/mlsibiochips/, 2012.

[18] L. M. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for Construction and Analysis of Systems*, pp. 337–340, 2008.