Matrix for Chemical IT (MATCHIT) 2013

Project no. 249032

Deliverable Report D6.4

Integrated production/computation IT Architecture for MATCHIT

Harold Fellermann, Carsten Svaneborg, Steen Rasmussen
FLinT Center for Fundamental Living Technology, University of Southern Denmark, Denmark

Uwe Tangen, Thomas Maeke, John S. McCaskill
BioMIP Center for Biomolecular Information Processing, Ruhr University Bochum, Germany

Omer Markovitch, Doron Lancet, Daniel Sorek
Weizmann Institute of Science, Department of Molecular Genetics, Israel

Benny Gill, Uri Shabi, Ehud Shapiro
Weizmann Institute of Science, Department of Biological Chemistry, Israel

Rudolf Füchslin, Matthias Weyland
University Ca' Foscari, European Center for Living Technology, Italy

Abstract

MATCHIT proposes a platform for integrated chemical production and information processing based on encapsulation in super-molecular chemtainers decorated with short DNA tags for chemtainer–chemtainer and chemtainer–matrix interactions. Chemtainers are embedded in programmable MEMS hardware that enables chemtainer manipulation. The resulting interplay of autonomous molecular computation and external electronic chemtainer manipulation enable programmatic setup of complex reaction cascades that allow for automated chemical production of desired target molecules from a limited set of chemical resources. MATCHIT presents a computational framework that spans from physical simulations of its molecular constituents to formal domain specific calculi for automated program inference and heuristic algorithms for yield optimization of target compounds. To close the loop back to the real world, we propose a semi-automatic mapping of control algorithms derived by idealized formal calculi back to machine code that operates MEMS hardware.

Introduction

Living systems integrate molecular recognition and information processing with material production on the molecular scale. The predominant locus of this integration is the cytoplasm, where a multitude of biochemical reactions proceeds simultaneously in separate mobile reaction compartments. Within the MATCHIT project, we have created the framework for an artificial cellular matrix that integrates information processing and material production in much the same way as its biological counterpart.

The MATCHIT framework employs super-molecular containers (termed *chemtainers* in the following), to mimic the topological organization of the cytoplasm where a multitude of simultaneous chemicals is organized in addressable compartments, and where transport and fusion of compartments triggers reactions of previously separated chemicals. As chemtainers, MATCHIT utilizes artificial vesicles, oil droplets, water droplets in ionic liquids, and DNA nano-cages. In the computational framework introduced here, the specific physical nature of chemtainers is mostly abstracted away. Chemtainers are decorated with short DNA tags that serve as programmable addresses and allow for chemtainer–chemtainer and chemtainer–matrix interactions through reversible hybridization.[1,2] DNA tags open up for DNA computing operations akin to the "key-lock" information processing mechanism found in

protein interactions.[3–5] This form of molecular information processing is governed by autonomous chemical reaction kinetics and allows for tight integration of chemical production and information processing. In particular, MATCHIT employs the DNA computing operations of address relabeling and simple Boolean computations.[6]

Whereas natural cells employ genomic information to regulate the resulting material production network, MATCHIT uses programmable electronic control. This is achieved by embedding chemtainers into a MEMS matrix that enables electronic control of chemtainer-chemtainer, and chemtainer-matrix interactions. In addition, special microfluidic channels allow for chemtainer creation, chemtainer docking, electrophoretic separation and re-encapsulation of cargo. Paired with real time feedback, this allows for control of chemical reaction cascades at the molecular level.

Taken together, the MATCHIT framework provides a platform for integrated chemical production and computation on the micro-scale: the interplay of autonomous molecular computation and external electronic chemtainer manipulation should enable the programmatic setup of complex reaction cascades that allow for automated chemical production of a desired target molecule from a limited set of chemical resources.

Integrated Chemical Production and Computation

Information technologies are an integral part of the MATCHIT project. Last but not least, electrodes of the MEMS hardware have to be controlled by some software. This, however, is only one aspect of the computational architecture underlying MATCHIT. Just as programming traditional IT with machine language is too tideous a task, we expect the explicit allocation of electrode voltages to be too cumbersome to be applicable in praxis. This motivates the development of a high-level MATCHIT language that abstracts away MEMS hardware specifications by offering data structures, commands and control structures that formalize the logical organization of chemical states and their transitions. As this language will to some extend formalize the structure and dynamics of chemical reactions and supermolecular organization, it will have to integrate domain knowledge from physics and chemistry. In this regard, we found it beneficial to develop physical models of chemtainers and tags with which we can study the detailed dynamics of integrated, externally driven tag-chemtainer systems. Tools from computer science are further used to derive and analyze control strategies that synthesize a desired target molecule, optimally with high yield. Eventually, one aim of the computational MATCHIT

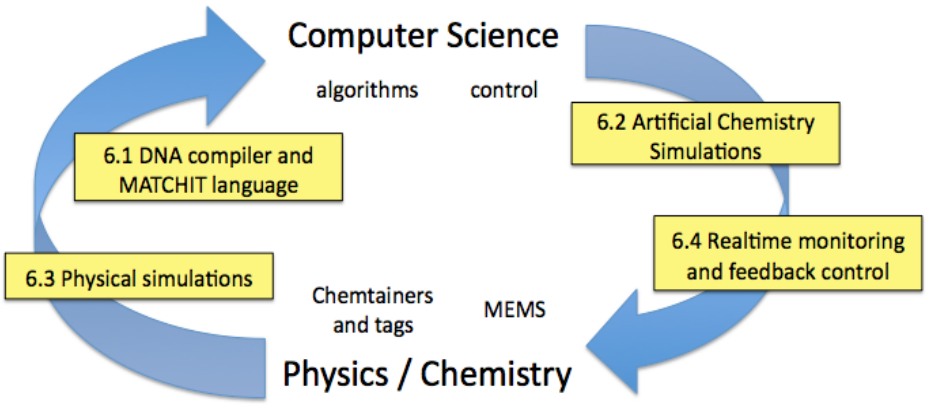

Figure 1: Physical simulation is used to obtain insight into detailed chemtainer–tag–matrix interactions. Their results feed into the development of a MATCHIT language that captures chemical arrangement and transitions in a formal system. Computer science tools are used to derive control strategies for MATCHIT productions in artificial chemistry simulations. Finally, these control algorithms are fed into a real-time feedback control software used to steer the MEMS hardware.

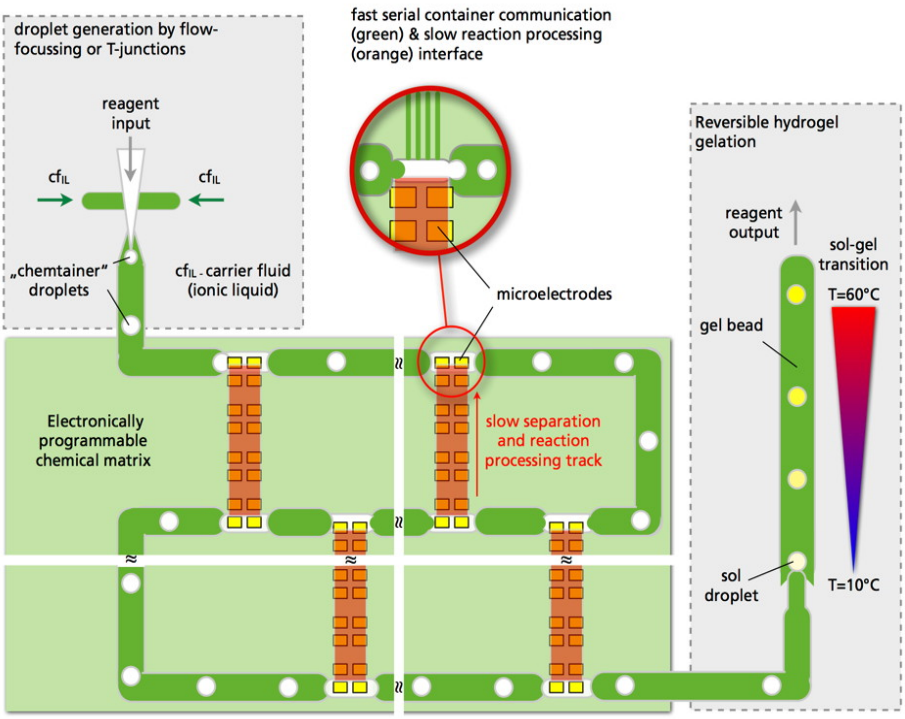

Figure 2: Initial target microfluidic architecture for distributed chemtainer processing, as outlined in the proposal. (see text). The chemical matrix is simplified to a pseudo 1-dimensional system (green) with cross-connections (vertical, orange) for content manipulation and product cleanup.

architecture is to programmatically infer control algorithms of the MEMS hardware from a given molecular target structure. We call such a program a *compiler* for MATCHIT. The entire organization of the computational MATCHIT architecture is summarized in Figure 1.

The initial target microfluidic architecture proposed in the project to approach this chemical matrix is shown in Figure 2. It shows the processing of chemtainers (e.g. droplets or vesicles) flowing in a linear channel meander, with vertical selective active transport segments enabling integrated separation and cyclic processing on chemtainer contents. In the project, this architecture was generalized to include multiple chemtainer channels, combinatorial creation of droplet chains and other features, but the basic principle was retained.

Physical Simulation

As MATCHIT operates over chemicals and supermolecular aggregates, it is important to understand in detail the physical properties of involved material, namely chemtainers, tags, and their cargo. Next to experimental expertise, we rely on computational physics to give us insights into the rather intricate dynamics of chemtainers and DNA tags. Importantly, these simulations simulation provide an added layer of physical detail and information, that can form a realistic molecular foundation for describing MATCHIT configurations at more abstract levels.

We have developed physics based molecular models of (oil-droplet) chemtainers and DNA tags, used to study chemtainer formation and dynamics as well as tag-tag and chemtainer-tag interactions.[7] The model includes the physics of chemtainer self-assembly aspects, tag anchoring, and the sequence

specific dynamics of tag addresses. It allows us to perform dissipative particle dynamics studies of chemtainer dynamics at the molecular length scale including hydrodynamic effects from the MEMS environment. Molecules are represented as strings of soft beads joined by springs. Unfavorable interactions between beads representing e.g. water, oil, surfactants and tags with hydrophobic anchors drive their self-assembly into tagged surfactant covered oil droplets. Beads representing DNA nucleotides have specific interactions modeling Watson-Crick pairing and can be dynamically introduced and broken e.g. by heating the system above the DNA address melting temperature.

Physical simulations have been used to study kinetic details of DNA tag interactions, where we have analyzed chemtainer addressing, and address relabeling (Figure 3), as well as DNA computation based on DNA strand displacement (Figure 4). In particular, we have used the physical simulation framework to elucidate the performance and fidelity of AND gates similar to the DNA computing gates underlying MATCHIT.[8,9] Equally, our simulation framework can be used to study chemtainer-tag interactions such as chemtainer fusion by means of complementary DNA addresses (c.f. Figure 5).

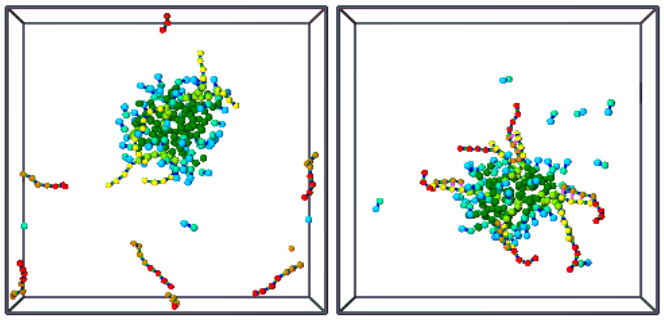

Figure 3: DPD simulation of chemtainer relabeling physics (M6.1). The chemtainer here is a surfactant covered oil droplet labeled by 6 tags in the presence of 6 relabeling constructs. Oil is shown as dark green beads. Surfactants head and tail groups are shown as light green and light blue beads, respectively. The tags consist of an address part (small yellow beads) and a hydrophobic anchor (small bright green beads). The relabeling constructs consist of an anti-address (small orange beads) complimentary to the tag address, and a new address (small red beads). Initially the relabeling constructs are all in solution and the chemtainer type is given by the yellow addresses (left). During the simulation each relabeling construct finds a tag to bond with and in the final state (right) all tags are relabeled with red addresses. Note furthermore that periodic boundary conditions apply to the simulations and renderings, e.g. constructs escaping the bottom of the box reenter from the top of the box. Water beads are present but not shown.

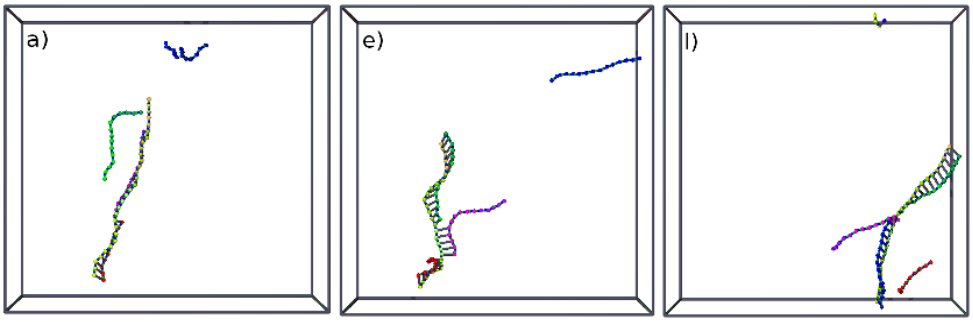

Figure 4: Snapshots from the DNA strand displacement simulation of an AND gate. Initially, the system consists of two sSDNA strands that codify input signals (green and blue) and a dsDNA complex where a long strand (yellow) hibridizes two shorter DNA strands (magenta and red), but also exposes a short toehold region complementary to green (left). The green input strand binds to the gate and subsequently releases the magenta strand which is slightly longer than the input and thus exposes a second towhold region (center). Eventually, the blue input strand binds to the gate and displaces the red output strand (right). Once the red strand is released, the kinetics of the gate is essentially irreversible.

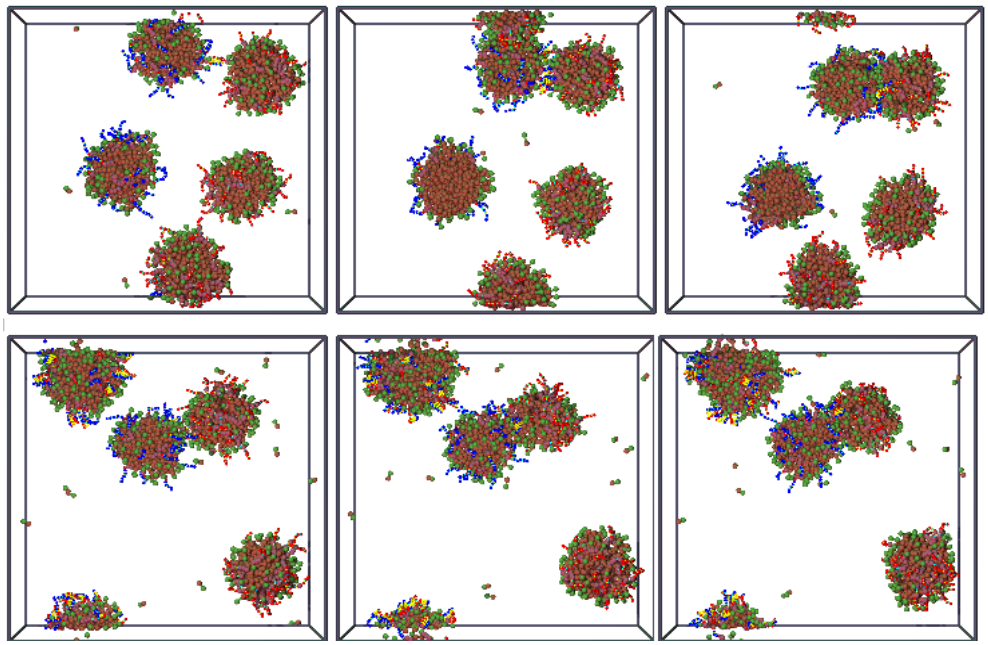

Figure 5: Six snapshots from two Dissipative Particle Dynamics simulations of freely floating surfactant covered oil droplets. The droplets are labeled by complementary ssDNA tags (blue, red) tethered to the droplets by an anchor. Hybridization bonds are shown as bright yellow. In the simulation shown in the top row, tag complementarity is from tip-to-anchor and tip-to-anchor. The dsDNA tags are observed to a form long lived bridge between the two chemtainers, which eventually leads to fusion. In the simulation shown in the bottom row tag complementarity is tip-to-tip and anchor-to-anchor. In this case, the dsDNA tags pulls the two anchors together, which induces a very fast fusion between the chemtainers.

A Formal Calculus for MATCHIT

We have developed a formal language that can express possible arrangements of chemtainers, cargo, and DNA tags in a symbolic representation of the MATCHIT MEMS.[10] Words of this language correspond to some well-defined physical state of the microfluidic device, an example being shown in Figure 6. States can be any composition of arbitrary molecules, DNA tags, DNA gates, and chemtainers which are possibly decorated with DNA tags and hold arbitrary cargo. Each item of the state is situated at a specified location. Formally, the language is defined through the following recursive grammar (where the broken vertical bar signifies choice):

$$
\begin{aligned}
\text{global state} \quad & \mathsf{S} := \emptyset \;\mid\; \mathsf{S} \circ \mathsf{S} \;\mid\; x_i : \mathsf{P} \\
\text{local state} \quad & \mathsf{P} := 0 \;\mid\; \mathsf{P} + \mathsf{P} \;\mid\; \mathsf{t}\,(\!|\, \mathsf{P} \,|\!)\, \mathsf{t} \;\mid\; \mathsf{s} \;\mid\; m_j \\
\text{tag set} \quad & \mathsf{t} := \diamond \;\mid\; \mathsf{t}|\mathsf{t} \;\mid\; \mathsf{s} \\
\text{tag} \quad & \mathsf{s} := s_k \;\mid\; s_k^{\top} \;\mid\; \mathsf{t} \rhd \mathsf{s}
\end{aligned}
$$

In this representation, the global MEMS system state is a discrete set of locations x_i each of them having a local state P. Each local state can either be empty (0) or a composition (+) of molecules m_j, tags s, and chemtainers (indicated by half-moon parentheses) that are decorated with some tag set t (diamonds are used to denote empty tag sets and a vertical bar is used for the composition of tag sets). Tags, in turn, are either single DNA strands that are potentially complementary to other tags (indicated by the marker $^{\top}$), or they can be DNA join gates (indicated by a triangle arrow) that release the tag s in the presence of all input tags t.

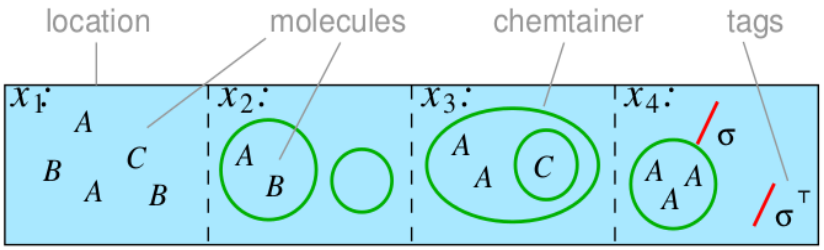

Figure 6: An example MATCHIT state that can be denoted by a word of the MATCHIT language.

On top of this grammar, we define transitions that reflect state changes either due to autonomous chemical kinetics or external programmatic control. An example of the first is the operation of a DNA AND gate that binds two input strands to release and output strand (compare to Figure 4). In the MATCHIT language, this can be written in the following way:

$$\sigma + \tau + \sigma^\top | \tau^\top \rhd \rho \longrightarrow \tau + \tau^\top \rhd \rho \longrightarrow \rho$$

The initial system consists of two input strands σ and τ, and a gate that releases ρ in the presence of bothinput strands. With the binding of σ the system transitions into the second state, where only the input strand τ and the partly consumed gate remains. Eventually, the remaining gate binds the input τ and and the state transitions into the output ρ. Inactive elements, such as the consumed gate, are not captured by the language.

An example of non-autonomous transitions is fusion of complementary tagged chemtainers (which we assume to be triggered by modifying the chemical environment):

$$\mathbf{fuse}(x): \qquad x : \sigma \langle\!(P)\!\rangle + \sigma^\top \langle\!(P')\!\rangle \longrightarrow x : \langle\!(P + P')\!\rangle$$

Unlike the previous transition, the fusion only occurs when the environment, the MEMS controller, initiates a fuse operation. A set of such controller inducible transitions forms the instruction set of a programming language that we have devised to denote control algorithms of the MATCHIT device. This programming language, which features sequential, parallel, and conditional execution is defined by a stochastic semantics that allows us to derive how a certain program transforms some given state of the MATCHIT device.

We have proposed a set of eight elemental instructions in order to feed chemtainers or tags, bind tags to chemtainers, move or arrest chemtainers and tags, fuse or burst chemtainers, or wrap arbitrary content into chemtainers. We have proven that these instructions are sufficient to generate any desired system state programmatically, and we have developed an algorithm that derives a program that is able to produce a given target state.

Programmable Chemistry and Yield Optimization

While it would be possible to instantiate the general MATCHIT hardware for synthesis of one specific goal compound, the true benefit of MATCHIT lies in offering programmable means for the synthesis of a whole range of diverse chemical compounds. As an example of such programmable chemistry, we have studied the class of branched oligomers, such as oligosaccharides. The synthesis of complex branched molecules is still a demanding task. One of the important issues is controlling potential side

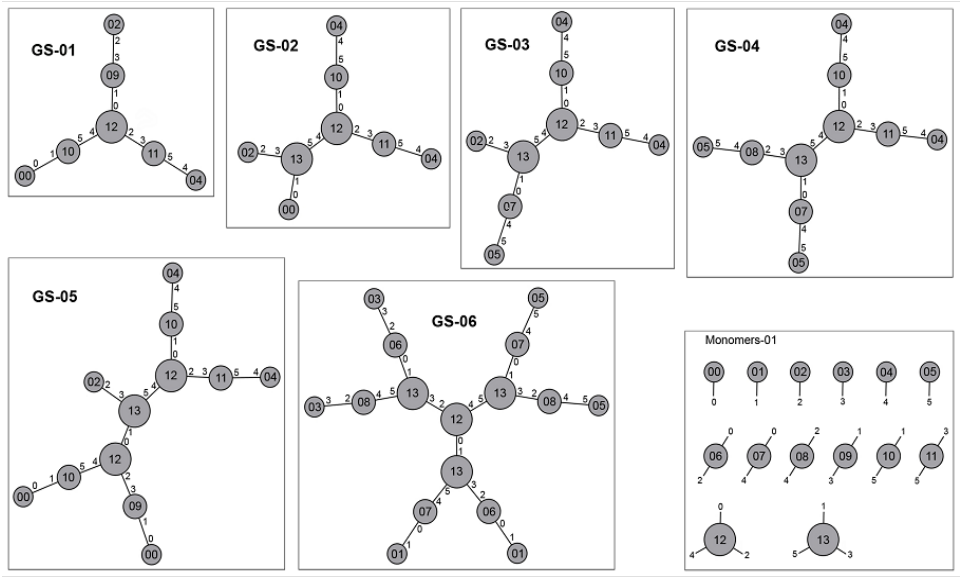

Figure 7: Various goal structures synthesized in spatially heterogeneous microreactors. Each structure is composed of some of the building blocks (monomers) given in the inlet in the lower right corner. A monomer is equipped with up to three linkers, making it possible to establish a link between linker pairs 0-1. 2-3 and 4-5. During synthesis, monomers or compounds of monomers can be coupled, as long as a pair of linkers is matching and the chemical environment is such that the coupling can be established.

reactions.[11] Whereas the synthesis of a linear chain molecule can happen in a template-based manner, and the according reaction – at least in theory – leads to an unambiguously defined end product, no comparable technique is available for branched structures.

Given the many types of chemical bond available in the lab, (e.g. so called click-chemistries[12]), building arbitrary branched structures is not a significant problem in itself. The challenge is controlling the reaction pathways. In our study, we demonstrated *in silico* that it is possible to create a spatially heterogeneous reaction environment with a bias towards specific reaction channels that increases the yield from the reaction.[13] Importantly, this increase is purely due to spatial organization and happens without any kind of interference with the kinetic constants for the reactions. Briefly, we combined two self-assembly processes: first, we used self-assembly to construct a two-dimensional micro-reactor from chemtainers (see details below). Second, we used the micro-reactor as a platform for the self-assembly of branched oligomers from monomeric building blocks. In what follows, we will limit our discussion to the spatial structure of the micro-reactor and the way we exploited it to modulate the self-assembly process.

The self-assembly processes we employed both use selective linkers. Both are reversible (the importance of reversibility is discussed in Whitesides & Boncheva[14]). Our primary aim was to synthesize branched goal structures as shown in Figure 7. Goal structures are composed of monomers, each equipped with up to three linkers. As described in the figure caption, we used three types of linker. Whether or not the bonds determined by these linkers can be established depends on chemical conditions in the environment, e.g. the presence of catalysts and whether the linkers match. The reactions required to match linkers take place in chemtainers: potentially microscopic reaction containers that can be linked to other chemtainers to build spatially heterogeneous reaction environments. In the first self-assembly process, the chemtainers self-assemble to form the reaction environment for the second self-assembly process, namely the formation of the branched molecules (the goal structures).

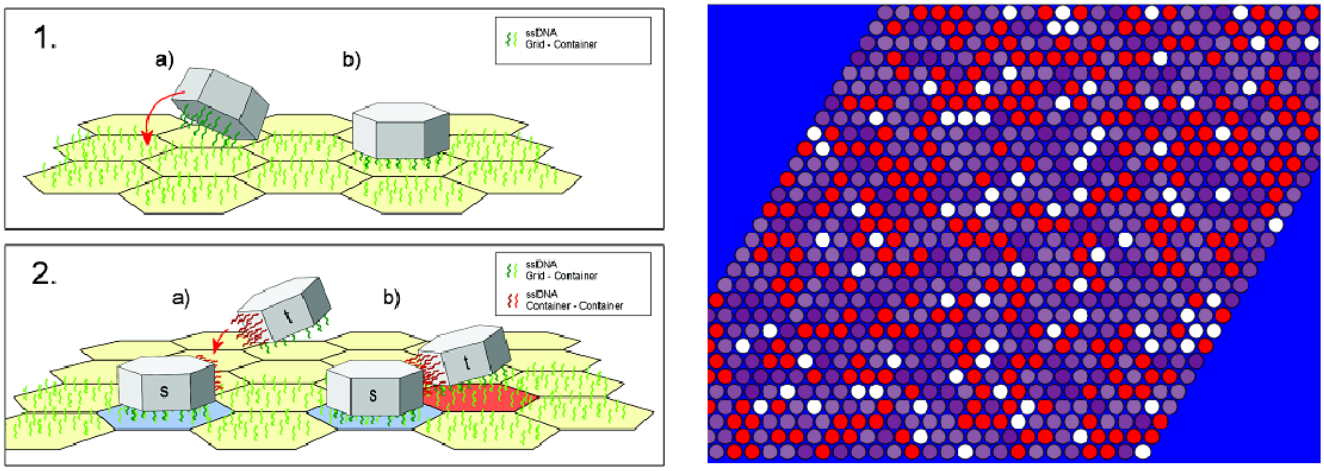

Figure 8: left: Self-assembly of the 2D grid of containers via specific linker molecules (ssDNA). The upper part (1) shows the association of a container to the grid, (2) illustrates the specific association between two containers (s, t). Right: example of a spatially heterogeneous micro reactor. A circle indicates a single container and the colors the different container types.

To obtain a spatially heterogeneous reaction environment, we assumed hexagonal chemtainers of various types whose type was specified first by their internal chemistry (to be discussed below) and second by selective linkers on their edges (e.g. DNA-based addresses). The chemtainers self-assemble on a two-dimensional substrate. Their bindings to the substrate are non-selective and relatively weak. Where they are available, bindings to neighbors are reversible and selective. The chemtainers are modeled as freely floating above the substrate, binding to the ground and to already bound adjacent chemtainers. For an illustration see Figure 8. Bonds between chemtainers are reversible: a bound chemtainer can be released again. If a chemtainer is connected to six matching neighbors, the total binding energy is tuned so that that release is no longer probable. The self-assembly process is run long enough to ensure the emergence of a structure with a defined neighborhood correlation, though not necessarily a spatially regular pattern (Figure 8, right).

Different types of chemtainers provide chemical environments capable of catalyzing the establishing or breaking of some of the possible links between the molecular monomers depicted in Figure 7. Mono- and oligomers are assumed to diffuse between chemtainers. Despite the diffusive processes, the functionality of the chemtainers is maintained. One way of realizing this is by anchoring catalysts to the walls of the chemtainers. If these walls are scaffolds with sufficiently large holes, oligomers can be transported via ordinary diffusion.

If one uses only a small number of different types of linkers (a realistic assumption in glycochemistry) the chemical functionality of the chemtainers is determined exclusively by the type of link and not by the molecules that are linked. However, it is also possible to link two oligomers if their respective linkers match. This is what makes it so difficult to control the process: the limited number of linkers means that the combination of intermediates (oligomers or monomers) is not sufficiently specific, and that many side reactions are possible.

In a one-pot reaction, all oligomers may be connected, always provided that they are equipped with matching linkers. However, in our system, the spatial arrangement of the chemtainers implicitly biases some reaction channels, leading to an increase in yield. For example, our method offers a ten-fold increase in yield for the synthesis of oligomer GS-04 (see Figure 7). In Figure 9 we show the results of an evolutionary optimization of the properties of the chemtainer. As explained above, chemtainers are characterized first by their linkers and second by their functionality. It was these properties we

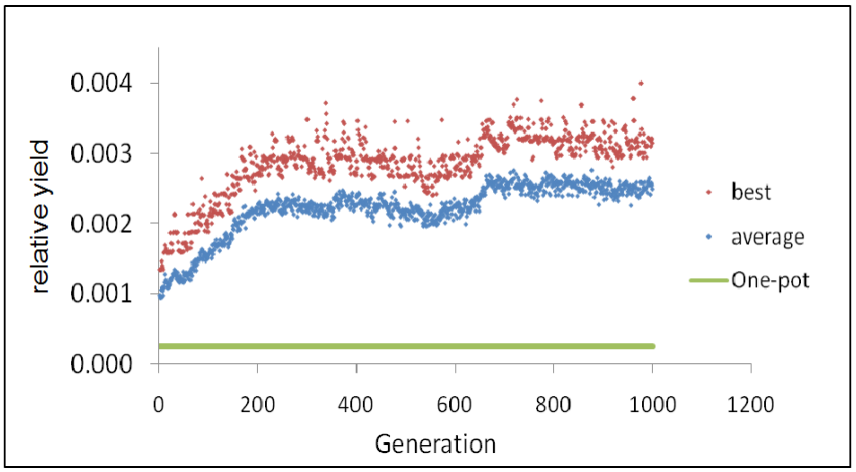

Figure 9: Development of the yield of the goal structure GS-04 during 1000 generation of the evolutionary algorithm. The lower line indicates the expected yield of polymerization in a homogeneous reaction environment. Chemtainers are characterized first by their linkers to other chemtainers on their edges and second by their chemical functionality, namely the ability of their content to influence the links between the building blocks of the oligomers to by synthesized. We evolved these properties using an evolutionary algorithm. For each combination of properties, we performed 20 complete simulations of the self-assembly process.

attempted to optimize. For each set of parameters, we performed 20 runs, each simulating the self-assembly of the spatially heterogeneous micro-reactor and the subsequent synthesis of the goal oligomer (GS-04). We then compared the results with those for a one-pot reaction. Figure 9 shows the average yield (as percent of the total number of molecules synthesized) and the yield of the best run. Interestingly the best result was not far from the average. This indicates that our procedure is rather robust.

The arrangement of the chemtainers biases certain reaction pathways by enhancing the probability of some sequences of additions or removals of parts of an intermediate molecule. As an example, consider a chemtainer with functionality A, surrounded by chemtainers with functionality B and C, but not D. When an intermediate molecule M reaches chemtainer A, it undergoes reactions in the order A-B or A-C but not A-D, except for the case that M travels through chemtainers B or C unaffected and somehow reaches a chemtainer with functionality D. In a spatially homogeneous environment, it would be possible to obtain the same effect by varying kinetic constants and reaction rates with respect to time, perhaps by transferring intermediates from one chemical environment to another. However, this transfer would require external control. The system we used here makes it possible to remain homogeneous with respect to time and to delegate the creation of specific, temporarily ordered sequences of reaction environments to the interplay between diffusion and the patterned reaction environment.

System Level Simulations of the MATCHIT System

The MATCHIT automaton (MA) is an abstract simulation of the MATCHIT system.[15] Chemtainers are fed into a tube divided into segments, and flow along the tube until they reach an output chamber (see Figure 10). Each chemtainer has a *tag* and content. Chemtainers of complementary tags will fuse with each other. The initial content is derived from a predefined chemistry, once chemtainers fuse their contents mix and react according to the chemistry rules, if at all. The segments of the tube also have tags and chemtainers that arrive at a segment with a complementary tag will be delayed at that segment

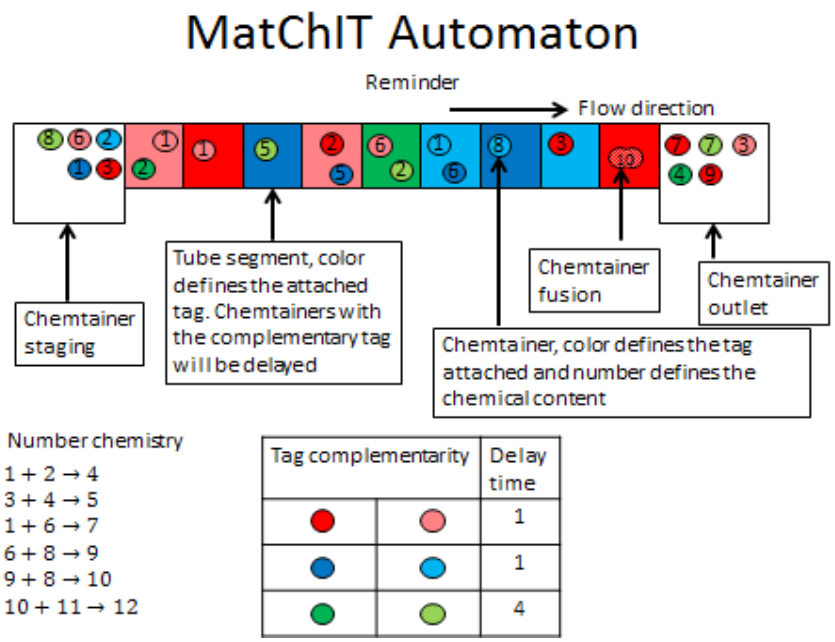

Figure 10: The MATCHIT-automaton: Chemtainers are transported from an input chamber (left) to an output chamber (right) through a tube that is decorated with DNA tags (colors). Chemtainers are equally equipped with tags, and complementary tags produce chemtainer-matrix interactions which results in delayed transport. If complementary tagged chemtainers meet, they fuse and mix their contents (indicated by numbers), thereby potentially initiating chemical reactions.

for a predefined number of time steps. If two chemtainers arrive at the same segment and their tags are complementary they fuse and their content mixes. If the content matches one of the predefined reactions, the content will be replaced by the product. In the case of two competing fusion events the fusion is decided randomly. We have also implemented DNA AND gates in MA, which allow a chemtainer to change its address after fusion, indicating that a certain reaction step inside has occurred.

The automaton model has been used to study several artificial toy chemistries, the branched oligomer chemistry presented in the previous section, as well as a template directed DNA amplification chemistries investigated by experimental groups of the MATCHIT consortium.

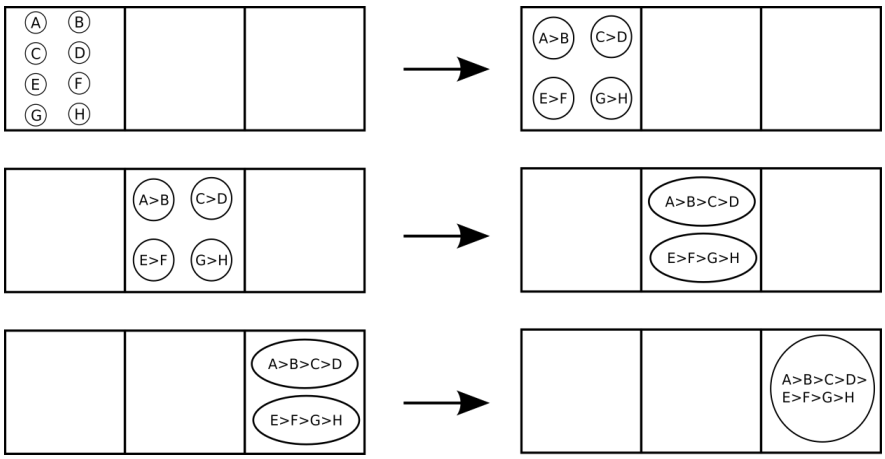

Figure 11: The synthesis of a linear octomer "A>B>C>D>E>F>G>H" using the MATCHIT-automaton. Inside each cell, two chemtainers are fused into one, leading to a polymer twice as long as its predecessors. Hence, the polymer can be synthesized in log2(8)=3 steps.

We showed that the time required for the synthesis of linear polymers in MATCHIT–automaton can be substantially reduced. While classical approaches take n steps to synthesize a linear polymer consisting of n polymers, the MATCHIT–automaton can achieve the same task in $\log_2(n)$ steps using the principle depicted in Figure 11.

We have also investigated evolutionary algorithms as heuristics to find MA control structures for high yield of target compounds. In this regard, the MA tube vector (tagging of each tube site) was subject to evolutionary optimization, while the set of input chemtainers has been kept constant.

DNA Addresss Compiler and Design Checker

The compilers described in the last sections generate control programs which utilize a library of DNA tags. Generally, the algorithms employ a list of symbolic tag names with no concrete DNA sequence information. To actually execute a generated program, these symbolic names need to be translated into specific nucleotide sequences.

In our DNA address compiler and design checker, tags are generated by using an evolutionary algorithm: an initial group of ssDNA sequences are randomized. For each pair of primers a Needleman–Wunsch global sequence alignment score is computed.[16] It is assumed that alignment scores correlate with the tendency of primers to form undesirable dimers in solution. The pair with the highest score is chosen and mutated. Mutations are carried out by substituting a random base in one of the pair with a different base and re-computing alignment score. The first mutation found to lower alignment score is selected and preserved. Next, all scores are recomputed and again the pair with the highest score is mutated. The whole process is repeated for new groups of sequences until no further improvement is achieved over several hundred cycles, for each group. The group of sequences with the lowest sum of all scores is provided as an address library.

Control of the MEMS Hardware

In the course of the project the MATCHIT-processor, an example of which is shown in Figure 12, has developed to a spatially resolved register-machine for chemtainers. A general chemtainer processing language description has been designed, (and extended to a compiler specification for applications to DNA editing in another EU-project CADMAD). In MATCHIT this intermediate higher level language was required to ease mapping between the physical hardware of the MATCHIT-processor and the abstract domain of the MATCHIT-calculus and broaden the connection to the other abstract description schemes documented above.

The language defined is comprised of ways to use chemtainers as input and output and to use rules of different kinds inbetween to realize the processing of the chemicals or sub-chemtainers, like nano-containers or vesicles, in the chemtainers. The figure above assumes droplets as chemtainer realizations. The following features are designed into the language:

Input/Output-set

Input is separated into two different sets: (i) the set of available chemicals or materials, including the knowledge of all required physical and chemical properties and (ii) the set of chemical rules including timing, physical and chemical dependencies of and between the rules.

Chemicals can be either DNA sequences which have been created via the DNA address compiler or other substances like enzymes, special buffers, salts or nano-containers which are carried with the

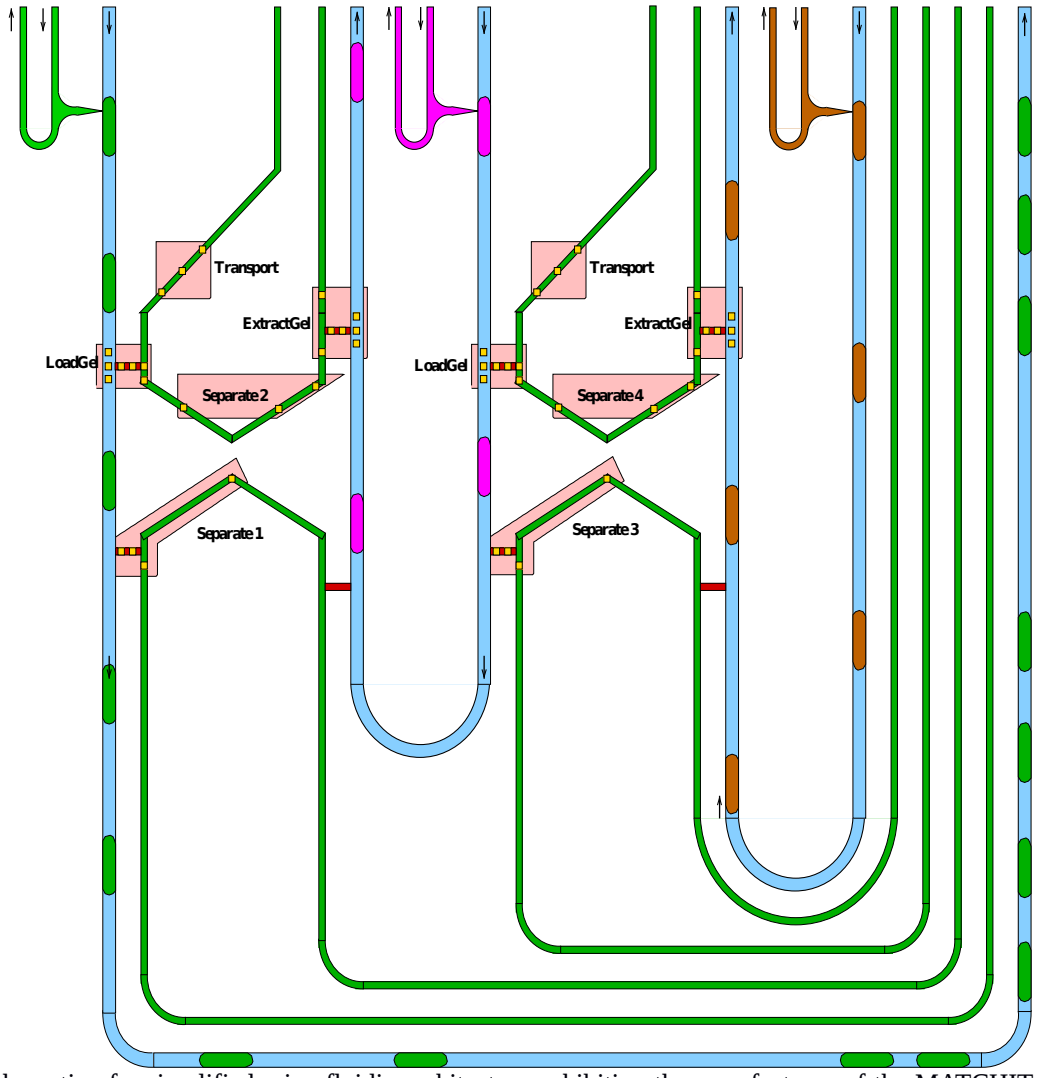

Figure 12: Schematic of a simplified microfluidic architecture exhibiting the core features of the MATCHIT-processor. 20 fluidic in- and outputs, including three droplet chains (light blue) created on chip and four separation tracks (green) allow a relatively flexible set of iterative processing operations to be performed.

chemtainers. The knowledge about the cargo in the chemtainers is divided into two parts: run-time data, like deployed quantities or boundary conditions and knowledge of the physico-chemical properties of the chemtainer ingredients and their interdependencies.

Output can be taken from modified input-set chemtainers (Output A) or from modified register-set chemtainers (e.g. Output B). It is assumed that these chemtainers can be captured in a tube and then, after freezing, cut out of the capillary.

Rules

Rules essentially map the algorithm into the spatially distributed system. These rules could e.g. have been derived by simulations done in the MATCHIT-automaton, MATCHIT-calculus or the other simulation environments. Three different types of rules are envisaged:

There are chemical rules, which can either contain chemical names or types, as in SBML,[17] reactions between chemicals can be encoded. These rules are written in canonical form and are active throughout the computation acting in parallel.

The canonical form follows the notation: A + B → C + D

Of course, there are not only bimolecular reactions possible. Without writing a symbol, emptiness is assumed (→ A means, A is created from nothing, which means insertion of a substance into a chemtainer, in the same manner A → means the extraction of a chemical or nano-container from a chemtainer). Rules are only executed, in the compiler internal model, when the detected concentration of the source-chemicals is above a certain threshold.

The second type of rules is a conditional form which resembles classifier-rules.[18] Whether a condition is fulfilled depends on the measured concentration or intensity, if the according chemical was specified with a suitable detection method. If no detection method is supplied a modeled assumption on the expected concentration is made.

A: execute rule X if concentration of A surpasses a certain threshold.

! A: execute rule U if concentration of A falls below a certain threshold.

A . B: execute rule Y if concentration of A and concentration of B are above a certain threshold.

A + B: execute rule Z if concentration of A or concentration of B are above a certain threshold.

The third class of rule types are basic operations, e.g.:

- loop V, which executes rule V *k* times.
- halt the execution of the system.
- wait, for a specific amount of time, at a specified temperature.
- mixture, a recipe to combine chemicals or nano-containers in a chemtainer.
- merge, *n* chemtainers are merged into a larger chemtainer, c.f. fuse()-operation in the calculus.
- select, a chemtainer from *k* chemtainers.
- sequence, process chemicals in a sequential manner.
- start and stop a specific chemtainer-chain.
- target, the chemical which should be delivered in an output-chemtainer.
- separate a specific sequence from the rest of the soup.

The language thus distinguishes equations ('→' is part of the command-line) from conditions (':' is part of the command-line) and function calls (curved brackets are part of the command-line). Only ‘sequence’ allows the specification of sequential processes, all other processes are working in parallel.

Register-set

The register-set is constituted via *n* lanes with empty chemtainers (these only contain buffer-solutions) to work as arbitrary storage places for chemicals or nano-containers. If the separate droplet lanes in Figure 12 are partially replaced by meanders, further registers can be added at alternating positions along the meandering lane of input chemtainers.

Processing

Whether the given task can be processed or not depends on the details provided by the rules and the constraints of the given chemicals. The compiler has to solve a satisfiability problem, which can be done exhaustively in simple systems and must be done by heuristics in more complicated endeavours.

Low-level operations

The following list comprises the low-level operations the compiler has at hand to satisfy the given constraints and rules:

- Start and stop of a chemtainer-chain, whether they are from the input-set or one of the many possible registers, they must be stoppable individually.
- Insert material into a chemtainer. The material must be affected by electric fields and it will be extracted from the gel-phase in the transportation and separation network, see an example in deliverables D5.3 and D5.4.
- Extract material from a chemtainer. The material must be affected by electric fields and it will be inserted into the gel-phase in the transportation and separation network, see an example in deliverables D5.3 and D5.4.
- Transport material along the transportation and separation network, if this material is sensitive to electric fields.
- Incubate material (wait), simply do nothing at a specific temperature.
- Divert a chemtainer-chain via a Y-structure and extract some chemtainers from the chain.
- Merge many chemtainers in a merging chamber (this is a possible extension of the compiler activities and is equivalent to the fuse()-operation in the MATCHIT-calculus).
- Split chemtainers with two consecutive chemtainer generators (this is a possible extension of the compiler activities).
- Separate material in the transportation and separation network. The separation will be done by online detection and assumes that two species will be separated and the slower species being the second one with a clear ditch in between the two separated species, see an example in deliverables D5.3 and D5.4.
- Temperature control to allow the implementation of PCR or optimal reaction conditions for the chemical substances.

The MATCHIT-IML (InterMediate Language)

The MATCHIT-IML is an interpreted and run-time compiled language that is intimately coupled to the MATCHIT-Ctrl software. It is especially developed to provide a higher-level control of the electrodes activation and the optical detection system. It allows one to incorporate arbitrary many scripts as low-level commands especially using level-one elements like state-machines, see Figure 14, which act as feedback-controllers. The most important task of these feedback-controllers is to abstract away the physical unknowns and details. The language is based on the following concepts:

The **instance** construct

In principle, an instance can be many things, a local region of solution containing diffusing particles or chemicals, a chemtainer (e.g., see Fig. 6), a gel bead or a single chemical compound. Its characteristics are specified by the properties added.

The **path** construct

Our main approach to chemical processing in microfluidic structures is that all chemicals are confined

to channels and electrodes are distributed along these channels. To reflect a network of channels the abstraction of a 'path' was invented. A path is a directed list of electrodes which happens to be in a consecutive sequence of fluid channels. The path is an arbitrary complex one-dimensional structure and should not incorporate mesh like structures to allow for a definite start-to-end walk. It is assumed that also chemtainers are following a prespecified path. A full set of operators is available for path manipulation: path, append, car, cdr, cons, cur, ele, last, move, next, nr, prev. For further constructs and information concerning the low-level details see the appendix.

In summary, we have constructed a hierarchy of descriptions that go from the very high level MATCHIT-calculus for the chemical matrix of chemtainers to an intermediate high level language dealing with chemtainer registers to the so-called intermediate language (MATCHIT-IML) which interfaces to the machine level language specific to the microfluidic implementation. The most developed and tested part of this hierarchy deals with the iterative process of content extraction, content separation and content injection from/into droplets. Further work will apply these achievements to real world chemtainer processing tasks, such as the microfluidic implementation of DNA editing as is being pursued in a follow on project CADMAD.

Tying the MATCHIT calculus and MATCHIT control

The compiler of the MATCHIT calculus, in conjunction with the yield optimization strategies, are used to find a general solution to a synthesis problem given by a specified target compound, resource molecules, and chemical reaction rules (canonical form, classifiers and basic operations). Ideally, the compiler would write out an intermediate script of low-level commands in MATCHIT-IML. This script can still and perhaps must be edited by a human operator in order to adhere to reaction and hardware specific boundary conditions that are not represented in the abstract frameworks. Examples of such

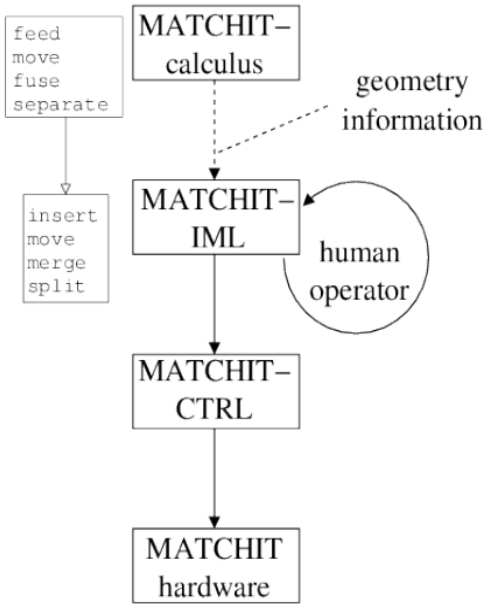

Figure 13: Workflow to create MATCHIT hardware machine code from programs generated by the MATCHIT calculus: the MATCHIT compiler generates a solution to the given synthesis problem only requiring the user to write a set of the three types of chemical rules (canonical form, classifiers and basic operations). The compiler would then write out an intermediate script of low-level commands which can be edited by the human operator. This script of low-level commands will then be translated by an assembler-stage which produces a script with all the detailed instructions to solve the high-level specified problem, see the ng_biopro-user-manual as a reference for available commands and available operations.

boundary conditions are temperature and pH control, or retention times. Likewise, the exact MEMS topology is currently not part of the formal calculi, and its abstract locations must be mapped onto concrete coordinates adhering to the hardware geometry (i.e. formalizations akin to Figure 6 must be mapped on a hardware architecture as depicted in Figure 12). Eventually, the post-processed IML script is translated by an assembler stage into a script with all the detailed instructions to solve the specified high-level problem, see the ng_biopro-user-manual as a reference for available commands and available operations. In general, we deem human interaction in this translation process to be necessary, since we have to map algorithms generated by an idealized calculus onto hardware control directives that have to function in the real world. Intermediate level MATCHIT-IML scripts are the appropriate place for this human interaction. The overall translation process as schematically shown in Figure 13.

Basic operations of the MATCHIT calculus have been mapped to the real microfluidic system as outlined in Deliverable 5.4. Furthermore, the MATCHIT calculus has been used to formalize the target implementation of iterative processing as a whole as presented in Deliverable 5.3 (Annex 2). These detailed applications, demonstrate the versatility of the language and its connection with the microfluidic system, and its ability to embrace new constructs.

Conclusion

MATCHIT has developed a platform for integrated chemical production and information processing based on encapsulation in super-molecular chemtainers decorated with short DNA tags for chemtainer–chemtainer and chemtainer–matrix interactions. Chemtainers are embedded in programmable MEMS hardware that enables chemtainer manipulation.

Physics based molecular models of chemtainers and DNA tags are used to study chemtainer formation and dynamics as well as tag-tag and chemtainer-tag interactions underlying DNA computing and chemtainer fusion. Information obtained from physical simulations and wet-lab experiments feed into the design of a domain specific formal language that can express possible arrangements of chemtainers, cargo, and DNA tags in a symbolic representation of the MATCHIT MEMS. We have devised a concurrent sequential process language to denote control algorithms of the MATCHIT platform, and we propose a set of eight elemental instructions for chemtainer manipulation. We have proven that these instructions are sufficient to generate any desired system state programmatically, and we have developed an algorithm that derives a program that is able to produce a given target state. As an example of such programmable chemistry, we have studied branched oligomers, for which we developed stratgeies for yield optimization based on hierarchical encapsulation and self-assembly characteristic for MATCHIT. We have analyzed both heuristic as well as analytic means of yield manipulation by means of symbolic system level simulations. In order to eventually steer the MATCHIT MEMS-hardware, we have constructed a hierarchy of descriptions that go from the high level MATCHIT-calculus for the chemical matrix of chemtainers to an intermediate high level language dealing with chemtainer registers to the so-called intermediate language (MATCHIT-IML) which interfaces to the machine level language specific to the microfluidic implementation.

In summary, MATCHIT presents a computational framework that spans from physical simulations of its molecular constituents to formal domain specific calculi for automated program inference and heuristic algorithms for yield optimization of target compounds. On the other hand, to close the loop back to the real world, we propose a semi-automatic mapping of control algorithms derived by idealized formal calculi back to machine code that operates MEMS hardware.

References

1. Hadorn, M. & Hotz, P. E. DNA-mediated self-assembly of artificial vesicles. *PLoS One* **5,** e9886 (2010).
2. Hadorn, M., Bönzli, E., Fellermann, H., Hotz, P. E. & Hanczyc, M. Specific and reversible DNA-directed self-assembly of emulsion droplets. *Proc. Nat. Acad. Sci. USA* **109,** (2012).
3. Seelig, G., Soloveichik, D., Zhang, D. Y. & Winfree, E. Enzyme-free nucleic acid logic circuits. *Science* **314,** 1585–8 (2006).
4. Zhang, D. Y. & Winfree, E. Control of DNA strand displacement kinetics using toehold exchange. *J. Am. Chem. Soc.* **131,** 17303–17314 (2009).
5. Qian, L. & Winfree, E. Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades. *Science* **332,** 1196–1201 (2011).
6. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R. & Shapiro, E. An autonomous molecular computer for logical control of gene expression. *Nature* **429,** 423–9 (2004).
7. Svaneborg, C. LAMMPS framework for dynamic bonding and an application modeling DNA. *Computer Physics Communications* **183,** 1793–1802 (2012).
8. Svaneborg, C., Fellermann, H. & Rasmussen, S. DNA Self-Assembly and Computation Studied with a Coarse-grained Dynamic Bonded Model. *Lect. Notes Comput. Sc.* **7433,** 123 (2012).
9. Svaneborg, C., Fellermann, H. & Rasmussen, S. Non-ideality and kinetics of a strand displacement AND gate studied with a dynamic bonded DNA model. *Nat. Comput.* (2013 in press).
10. Fellermann, H. & Cardelli, L. Programmatic control of nanoscale bioreactors. (2013 in preparation).
11. Koeller, K. M. & Wong, C. Complex carbohydrate synthesis tools for glycobiologists: enzyme-based approach and programmable one-pot strategies. *Glycobiology* **10,** 1157–1169 (2000).
12. Kolb, H. C., Finn, M. G. & Sharpless, K. B. Click Chemistry: Diverse Chemical Function from a Few Good Reactions. *Angew. Chem. Int. Ed. Engl.* **40,** 2004–2021 (2001).
13. Reller, B. Programmable self-assembling spatially heterogeneous micro-reactors. MSc thesis. University of Zurich (2010).
14. Whitesides, G. M. & Boncheva, M. Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proc. Nat. Acad. Sci. USA* **99,** 4769–4774 (2002).
15. M. S. Weyland, H. Fellermann, R. M. Füchslin, D. Sorek, D. Lancet. The MATCHIT automaton: Exploiting Hierarchical Compartmentalization. (2013 in preparation).
16. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48,** 443–453 (1970).
17. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19,** 524–531 (2003).
18. Holyoak, K. J. *Induction: Processes of Inference, Learning, and Discovery*. (MIT Press, 1989).

Appendix

Further constructs in the intermediate language are:

Task: All commands inside a task are executed in sequential order. Also blocks- and sub-blocks are treated sequentially. But arbitrary many tasks can be specified. These tasks are processed in a pseudo-parallel fashion. True parallelism cannot be used here because the single available camera has to move to the regions of interest (ROI, instead of the camera the xy-table is moving) and these ROIs might reside far apart from each other (in principle, the system could determine whether two or more tasks fit into the same field-of-view of the camera but this is a software-technically difficult task). This means that after a command finished in task A, task B is selected and the next command in that task is executed. These physical constraints have to be reflected in the physico-chemical knowledge of the compiler.

Variables: Variables are instantiated after first naming them in the assign command. Status-codes can be assigned to variables as well as to other variables. Basic arithmetic operations can be realized as part of the assign-command.

Block: A block starts with the begin-command and finishes at the end-command. A block can contain arbitrary many sub-blocks. All commands in a block are processed in sequential order. A block is a mere syntactical concept and has no physical counter-part.

Conditions: Three basic conditions are available, GT, GE and EQ. Comparing different types with each other is done implicitly. if supported, otherwise an error-message is issued. These conditions in many cases can be directly derived from the conditional expressions in the classifiers.

Control: Further typical commands of a structured programming language are: break, do, elif, else, enddo, endif, endtask, exec, if.

Low-Level: Some of the low-level commands to control the underlying hardware are:

nr_states: Number of states in automaton

add_state: A full state specification to a state-machine

move_back: Move xy-table to last position

move_ori: Move xy-table to original position

move_xy: Move the xy-table to the center of a given electrode.

set_pin: Control the polarity and activity of a pin or electrode.

set_sensor: Specify thresholds and other parameters

zoom: Zoom into a graphical window.

For illustration see the state-machine in Figure 14 (which has been used in the separation experiments, c.f. deliverables D5.4 and D5.3.

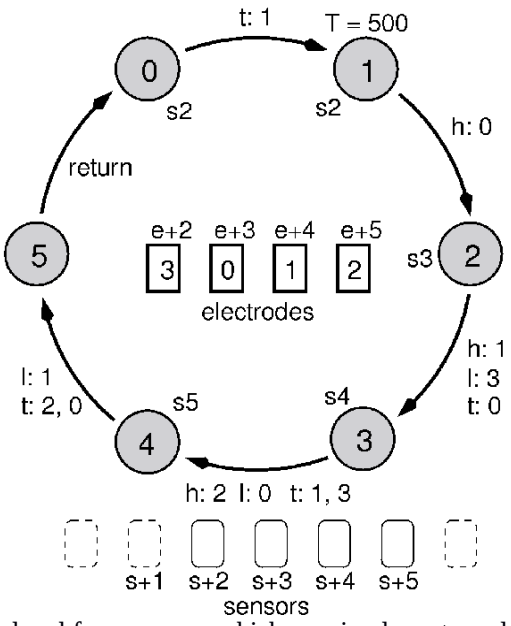

Figure 14: Four electrodes are activated and four sensors, which are simple rectangular areas in the camera image, are used to measure the current intensity of the fluorescence signal.

This state-machine is programmed into the controlling software. This software activates the electrodes, evaluates the camera image at the ROI and controls all remaining devices of the whole system. Because of the spatially distributed nature several state-machines can be connected in a chain-like manner, or moving along the path with the chemtainers, cooperatively solving the control-problem. The following script extract shows low-level commands used for the DNA-separation tasks:

```
eval_channel design/fluidic/biopro61_std/M1_B5_SEG.dat

control_add autoshift1 "general" r1833
control_add autoshift2 "general" r1834
     :
     :

#set_pin r114 high          # Activate PIN r114
#set_pin r1618 low   # Shielding with negative electrodes
#set_pin r718 low   # Shielding with negative electrodes
auto_shift_8 autoshift1 r1833 r1834 r1835 r1838 r1839 r1840 autoshift2
auto_shift_8 autoshift2 r1834 r1835 r1838 r1839 r1840 r1841 autoshift3
auto_shift_8 autoshift3 r1835 r1838 r1839 r1840 r1841 r1842 autoshift4
    :
auto_shift_8 autoshift10 r1840 r1839 r1838 r1835 r1834 r1833 autoshift11
auto_shift_8 autoshift11 r1833 r1834 r1835 r1838 r1839 r1840 separ1
separ1 separ1 r1833 r1844 r1845 autoshift12   # This is the separation step
auto_shift_8 autoshift12 r1843 r1842 r1841 r1840 r1839 r1838 autoshift13
auto_shift_8 autoshift13 r1842 r1841 r1840 r1839 r1838 r1835 autoshift14
auto_shift_8 autoshift14 r1841 r1840 r1839 r1838 r1835 r1834 autoshift15
auto_shift_8 autoshift15 r1840 r1839 r1838 r1835 r1834 r1833 autoshift1
```