Generic Strategies for Chemical Space Exploration

Jakob Lykke Andersen[1,4], Christoph Flamm[2], Daniel Merkle[1], Peter F. Stadler[2−7]

[1] Department for Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark [2] Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Wien, Austria. [3] Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Härtelstraße 16-18, D-04107, Leipzig, Germany. [4] Max Planck Institute for Mathematics in the Sciences, Inselstraße 22 D-04103 Leipzig, Germany. [5] Fraunhofer Institute for Cell Therapy and Immunology, Perlickstraße 1, D-04103 Leipzig, Germany. [6] Center for non-coding RNA in Technology and Health, University of Copenhagen, Grønnegårdsvej 3, DK-1870 Frederiksberg C, Denmark. [7] Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

Email: JLA:jlandersen@imada.sdu.dk; CF*:xtof@tbi.univie.ac.at; DM*:daniel@imada.sdu.dk; PFS:studla@bioinf.uni-leipzig.de;

*Corresponding author

Abstract

Background: Computational approaches to exploring "chemical universes", i.e., very large sets, potentially infinite sets of compounds that can be constructed by a prescribed collection of reaction mechanisms, in practice suffer from a combinatorial explosion. It quickly becomes impossible to test, for all pairs of compounds in a rapidly growing network, whether they can react with each other. More sophisticated and efficient strategies are therefore required to construct very large chemical reaction networks.

Results: Undirected labeled graphs and graph rewriting are natural models of chemical compounds and chemical reactions. Borrowing the idea of partial evaluation from functional programming, we introduce partial applications of rewrite rules. Binding substrate to rules increases the number of rules but drastically prunes the substrate sets to which it might match, resulting in dramatically reduced resource requirements. At the same time, exploration strategies can be guided, e.g. based on restrictions on the product molecules to avoid the explicit enumeration of very unlikely compounds. To this end we introduce here a generic framework for the specification of exploration strategies in graph-rewriting systems. Using key examples of complex chemical networks from sugar chemistry and the realm of metabolic networks we demonstrate the feasibility of a high-level strategy framework.

Conclusions: Graph grammars in conjunction with efficient, versatile exploration strategies are a powerful framework for combinatorial chemistry applications, allowing detailed investigations in very large chemical spaces, which is the foundation for understanding function of biological systems. The ideas presented here can not only be used for a strategy-based chemical space exploration that has close correspondence of experimental results, but are much more general. In particular, the framework can be used to emulate higher-level transformation models such as illustrated in a small puzzle game.

Background

The systematic computational exploration of chemical spaces have become topic of high practical relevance e.g. in drug design [1–4]. Recent efforts to gain insights into the distribution of properties in chemical spaces include the construction of large databases of hypothetical compounds. The "chemical universe database GDB-17" [5], for instance comprises 166.4 billion molecules of up to 17 atoms of C, N, O, S, and halogens covering the size and composition range of typical lead compounds. Beyond the diversity of molecules and their properties, however, potential synthesis pathways leading to them are a crucially important consideration in practice. This calls for methods to systematically explore chemical spaces in terms of restricted types of chemical reactions. Here we demonstrate how this can be achieved in a natural way by means of graph grammars in conjunction with efficient exploration strategies.

The *structural formula* of a chemical compound is a graph that represents the connectivity and mutual arrangements of its atoms. Atom types are given as vertex labels, while edges represent bond types. At this level of modeling, chemical reactions are naturally represented as graph transformations. Chemical reactions are explained and categorized in terms of *reaction mechanisms* that encapsulate the local changes of chemical bonds. In the formal framework of graph grammars, reaction mechanisms correspond to the productions (rules). Because of this conceptual alignment between chemistry and graph grammars, a variety of artificial chemistry models of different degree of chemical realism have been devised on this basis [6]. Of course, these purely combinatorial models of chemistry have their limitations. Deliberately disregarding the spatial embedding of molecules they cannot capture many aspects of stereochemistry and they are restricted to (over)simplified models of reactions energies and reaction kinetics. Graph grammar models are nevertheless of practical interest when the task is to explore large areas of chemical spaces and they provide a means of analyzing regularities in very large reaction networks.

Several graph rewriting tools have become available in the recent past, see [7] for an overview. Application areas beyond chemistry include model checking and verification, proof representation, and modeling control flow of programs among many others. A strategy language to control the application of graph rewriting rules has been presented in [8] for PORGY [9, 10]. A strategy framework for exploring chemical spaces has very different design goals from the properties desirable for applications within different areas of computer science. For instance, chemical graph grammar rule frequently merge or split graphs, since connected components correspond to individual molecules. Hence a chemically motivated component handling is required. The theoretically reachable chemical spaces can be infinite, e.g., when the rules and the starting material allow polymers to form. Exploration thus may not halt except due an enforced resource (size) limitation. Decisions on how to expand the space are usually heavily influenced by chemical properties or additional data sources. Furthermore, the goal of an analysis might also be motivated by a chemical question such as the detection of chemical subspaces or the need to find specific chemical transformation patterns.

In this paper different types of "chemistries" will be used to demonstrate the different aspects of the strategic construction framework for fractions of the chemical space. The Diels-Alder reaction [11], a sigmatrope cycloaddition reaction between a conjugated diene and an alkene will serve as an example for a combinatorial complex chemical space emerging from a single reaction rule. The formose reaction [12], which subsumes the formation of sugars from formaldehyde is used to explore the impact of changes in "chemistry" (i.e. the set of reaction rules) on the structure and complexity of the chemical space. HCN chemistry is used to illustrate how construction of chemical space can be biased with experimental data.

The outline of the paper is as follows. In the methods section we will first present the formal framework including the Double Pushout Approach. Secondly, partial rule applications which will be heavily used in a chemical setting will be described. Thirdly, general strategies to explore chemical spaces will be introduced. In the results section we will apply our framework to several before-mentioned complex chemical settings. Finally a puzzle game we be used to illustrate the generality of our approaches.

Methods

Formal Framework

Chemical Graph Rewriting with the Double Pushout Approach

Molecules are always represented by connected graphs. Chemical reactions, however, more often than not, involve two or more interacting molecules as their "input" (educts) and there is no guarantee that the "output" (products) is connected. Thus we have to consider graph transformations that operate on not necessarily connected graphs. More precisely, we regard a graph G here as a multiset $\{g_1, g_2, \ldots, g_{\#G}\}$ of its $\#G$ connected components. All graphs are simple, i.e., without loops and parallel edges. Double and triple bonds are viewed as edge labels rather than multiple edges.

Several abstract formalisms for graph transformation have been explored in the literature, see e.g., [13] for a detailed introduction. We found that the so-called Double Pushout (DPO) approach provides the most intuitive direct encoding of chemical reactions and the closest connection to the language of chemistry. A DPO transformation rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ consists of three graphs L, R and K known as the left, right and context graph, respectively, and two graph morphisms l and r that determine how the context is embedded in the left and the right graph. The rule p can be applied to a graph G if the left graph L can be found in G and some additional consistency conditions are satisfied. This is modeled by the requirement that there is a *matching morphism* $m : L \to G$ that describe how L is contained in G. Intuitively, the copy of L is replaced within G by R in such a way that the context K is left intact, resulting in the transformed graph H. This operation, the derivation $G \xRightarrow{p,m} H$, is described in the framework of category theory by the requirement that the following commutative diagram exists:

$$
\begin{array}{ccccc}
L & \xleftarrow{\;l\;} & K & \xrightarrow{\;r\;} & R \\
{\scriptstyle m}\downarrow & & {\scriptstyle d}\downarrow & & {\scriptstyle n}\downarrow \\
G & \longleftarrow & D & \longrightarrow & H
\end{array}
\tag{1}
$$

The derivation $G \xRightarrow{p,m} H$ implicitly define the intermediary graph D and the result graph H as well as morphisms $d : K \to D$ and $n : R \to H$ that fix how the context and the right graph of the rule are embedded in the intermediary and the result graph, respectively. In terms of molecules (connected components) we can write $\{g_1, g_2, \ldots, g_{\#G}\} \Rightarrow \{h_1, h_2, \ldots, h_{\#H}\}$.

In applications to modeling chemistry, several additional requirements must be satisfied. Conservation of mass and atom types dictates that the restrictions of r and l to the vertex sets (atoms) are bijective. Furthermore, m (and by extension d and n) are subgraph isomorphisms and hence injective. We note in passing that this guarantees the existence of a bijection $a : V(G) \to V(H)$ known as the *atom mapping*. In the DPO formalism, furthermore, the existence of an inverse production $p^{-1} = (L \xleftarrow{l} K \xrightarrow{r} R)$, corresponding to the reverse chemical reaction, is guaranteed. Some more basic properties of chemical graph grammars can be found in [14]. Fig. 1 shows an example of a chemical derivation.

Proper Derivations

Consider a valid derivation $\{g_1, g_2\} \xRightarrow{p,m} \{h_1, h_2\}$ and an arbitrary graph g'. Clearly, the derivation $\{g_1, g_2, g'\} \xRightarrow{p,m} \{h_1, h_2, g'\}$ is also valid because the images of m and n are contained in $\{g_1, g_2\}$ and $\{h_1, h_2\}$, respectively. The graph g' is irrelevant for the transformation. We call a derivation $G \xRightarrow{p,m} H$ *proper* if img $m \cap g_i \neq \emptyset$ for all $g_i \in G$. It is not hard to see that the inverse of a proper derivation is again proper. Throughout the following sections we will assume every derivation to be proper, unless otherwise stated.
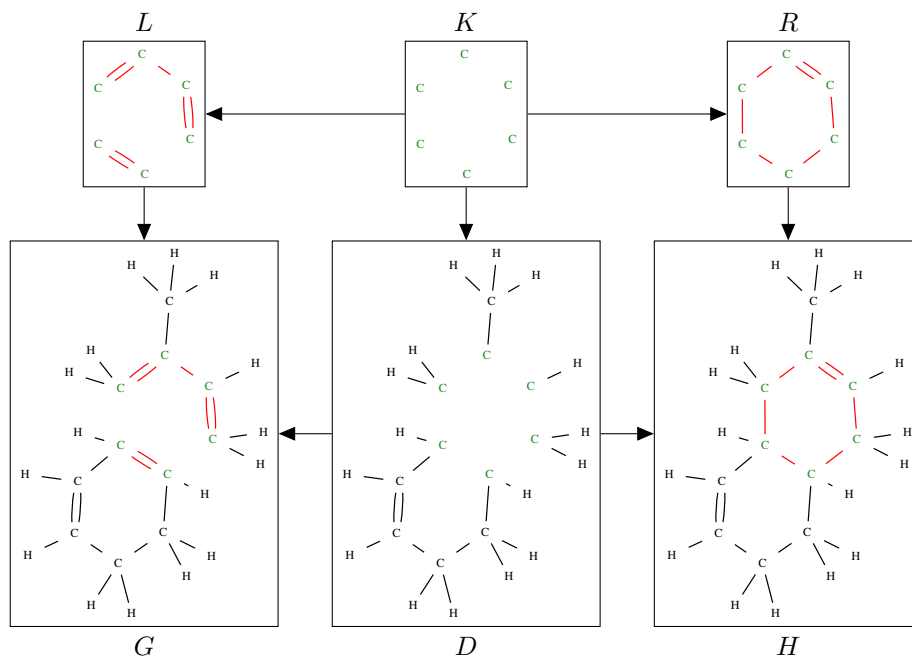
Figure 1: Example of a chemical derivation from cyclohexadiene and isoprene using a Diels-Alder transformation. The edges changed by the transformation is shown in red and the vertices from K are shown in green. Note that edges shown in parallel are in the underlying graphs a single edge with a special label to encode a specific chemical bond.
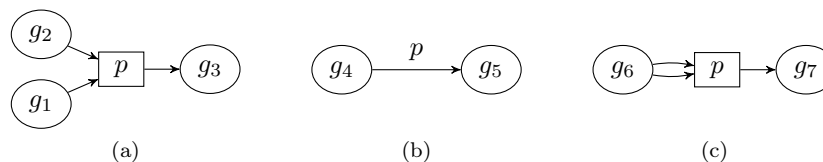

Figure 2: A bipartite graph notation for directed (multi-)hypergraphs, in which the production itself is drawn as a special type of intermediate vertex, is used in most cases; (a), $\{g_1, g_2\} \overset{p}{\Rightarrow} g_3$. We only make an exception for 1-to-1 transformations (isomerization reactions); (b), $g_4 \overset{p}{\Rightarrow} g_5$. Multiplicities are indicated by multiple arcs; (c), $\{g_6, g_6\} \overset{p}{\Rightarrow} g_7$.

Derivation Graphs

Chemical reaction networks can be represented as directed (multi)hypergraphs whose vertices are the molecules of the "chemical universe" under consideration and whose hyperedges represent chemical reactions [15]. Here, it is important to consider hyperedges as multisets to accommodate the stoichiometric coefficients, i.e., the multiplicities in which molecules enter a chemical reaction such as $2H_2 + O_2 \rightarrow 2H_2O$. Such networks can be constructed from experimentally observed data. An example is the Network of Organic Chemistry (NOC) [16–18], which shows a non-trivial organization concentrated around a core region of about 300 synthetically important building blocks and industrial compounds. Metabolic networks consist of the enzymatically catalyzed reactions constituting the chemical basis of modern life forms. They are available from dedicated databases, see e.g., [19].

In the framework of graph grammar models, an analogous *derivation graph* can be defined. Its vertex set consists of the connected labeled graphs $\mathfrak{G}$ that represent the molecules. Directed hyperedges connect the multisets $G \subseteq \mathfrak{G}$ and $H \subseteq \mathfrak{G}$ only if there is a proper derivation $G \overset{p,m}{\Longrightarrow} H$. The conventions for visualizing hyperedges adhere to the three examples in Fig. 2.
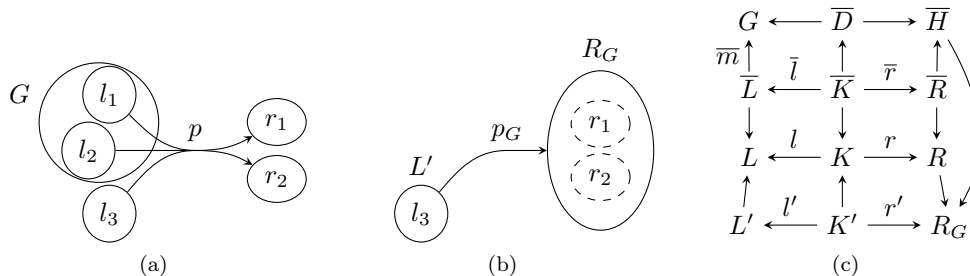
Figure 3: Partial application of some rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ to a graph G, with $L = \{l_1, l_2, l_3\}$ and $R = \{r_1, r_2\}$. The partial application is done through a partial matching morphism $\overline{m} : \overline{L} \to G$ with $\overline{L} = \{l_1, l_2\}$. The application results in a new rule, $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$ with $L' = \{l_3\}$, for which R is a subgraph of R_G. The transformed graph of G, called $\overline{H}$, is also a subgraph of R_G. Fig. (c) is the diagram of subgraph relations for a general partial rule application.

Transformation by Partial Rule Application

The core strategy to expand the underlying derivation graph is the discovery of new graphs by means of proper derivations implied by the direct application of rules. Given a rule $p = (L \leftarrow K \to R)$ and a set of graphs $\mathfrak{U}$, the task is to find all proper derivations $G \xRightarrow{p} H, G \subseteq \mathfrak{U}$ where G and H are multisets of graphs. This can be done by a testing of all k-multisubsets of $\mathfrak{U}$ for all $1 \le k \le \#L$. Since nearly all chemical reactions are mono-molecular or bi-molecular, we can restrict ourselves to $\#L \le 2$, at least when elementary reactions are of primary interest. Still, the number of multisets is $O(|\mathfrak{U}|^2)$. In the worst case, all unique multisets may give successful transformations, often leading to a combinatorial explosion that quickly becomes unmanageable. In the following section we show that a more detailed control of the multisets that are considered for transformation is desirable.

The key concept is partial rule composition [14], i.e., the binding of graphs to rules, resulting in partial rules that can be applied more efficiently in an exploration strategy. The idea is analogous to partial evaluation of functions by binding some of the variables. Full graph transformations are computed as repeated partial rule application in this framework. For the sake of brevity, we only sketch the idea here and omit a complete formal definition of partial rules.

A partial rule application of a rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ with $L = \{l_1, l_2, \ldots, l_{\#L}\}$ to a graph G, is a generalization of a full transformation of G in which only some but not all components of L do not match G. Thus L is partitioned into the matching part $\overline{L} \ne \emptyset$ and the non-matching remainder L'. The restriction $\bar{l}$ of $l : K \to L$ to the pre-image $\overline{K}$ of $\overline{L}$ defines the partial transformation rule $\overline{p} = (\overline{L} \xleftarrow{\bar{l}} \overline{K} \xrightarrow{\bar{r}} \overline{R})$. Using the restricted matching morphism $\overline{m} : \overline{L} \to G$ it can be applied to G resulting in graph $\overline{H}$. The remainder L' of L gives rise to a new rule $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$ whose right graph consists of the transformed version of G as well the original right graph R, i.e., it contains both $\overline{H}$ and R as subgraphs. A formal, diagrammatic representation is given in Fig. 3c. An abstract partial application is shown in Fig. 3a and 3b.

Given a not necessarily connected graph G and DPO transformation rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$, our task is to construct all partial rules obtainable by binding G to p. These partial rules can then be applied to further graphs, allowing for more efficient exploration strategies. The following algorithm enumerates these partial rules:

1. For all $l_i \in L$ find the set of all subgraph isomorphisms of l_i to G. That is, find $M_i = \{m \mid m : l_i \to G$ is a subgraph isomorphism$\}$ for $1 \le i \le \#L$.

2. For all nonempty subsets $\overline{L}$ of L, construct all partial matching morphisms, $\overline{m}$, by merging morphisms from each $M_j, l_j \in \overline{L}$. Note, that each $\overline{m}$ must be injective.

3. For each partial matching morphism, $\overline{m}$, apply p to G with $\overline{m}$ to obtain a new rule $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$.
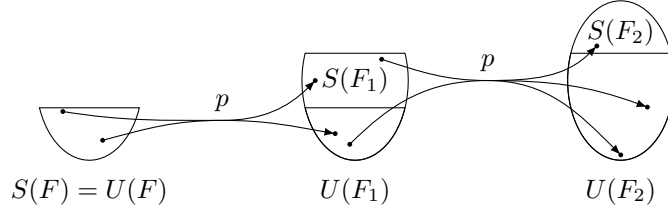
Figure 4: Illustration of the evaluation of $F_2 = p(F_1)$ for $F_1 = p(F)$ and some set of graphs, $S(F) = U(F) = \mathfrak{U}$. Each derivation must use at least one graph form the input subset. Two abstract derivations are shown with the endpoints indicating in which sets the graphs are.

The partial matching morphisms constructed from considering $\overline{L} = L$ are actually full matching morphisms, and so the resulting rule has $L' = K' = \emptyset$. In this case p_G represents the creation of R_G from an empty graph, and $G \xrightarrow{p, \overline{m}} R_G$ is a valid derivation. If G is connected, the derivation will additionally be proper.

In the following section we will regard a rule p as a function on sets of graphs, defined provisionally as:

$$p(\mathfrak{U}) = \mathfrak{U} \cup \bigcup_{\substack{G \xRightarrow{R} H \\ G \subseteq \mathfrak{U}}} H$$

That is, the result of applying p to a set of graphs, $\mathfrak{U}$, is $\mathfrak{U}$ itself along with all graphs derivable from $\mathfrak{U}$ using p.

Complex Graph States

Consider the problem of applying a rule p twice to a set of graphs $\mathfrak{U}$. That is, finding $\mathfrak{U}_2 = p(\mathfrak{U}_1)$ for $\mathfrak{U}_1 = p(\mathfrak{U})$. By our definition of rule application we have $\mathfrak{U} \subseteq \mathfrak{U}_1$, so when the algorithm described above is used for evaluating $p(\mathfrak{U}_1)$ it will find not only new derivations but also all derivations found when evaluating $p(\mathfrak{U})$. We therefore use a more complex state than simply sets of graphs. A *graph state* F is defined as a pair of ordered sets of graphs $(\mathcal{U}, \mathcal{S})$ with $\mathcal{S} \subseteq \mathcal{U}$. The elements, $\mathcal{U}$ and $\mathcal{S}$, will be referred to also as $U(F)$ and $S(F)$ respectively, where U and S are functions on the graph state. In the following we will denote $U(F)$ as the *universe* of the graph state F and $S(F)$ as the *subset* of the state. The order of graphs in the subset and in the universe is independent and is arbitrary unless otherwise stated.

We define the application of a rule p to a graph state F in the following manner. Let H' be all connected graphs derivable from $U(F)$ with p such that at least one graph from $S(U)$ is being transformed in each derivation:

$$H' = \{h \in H \mid G \xRightarrow{p} H : G \subseteq U(F) \wedge G \cap S(F) \neq \emptyset\} \tag{2}$$

The result $F' = p(F)$ is such that

$$U(F') = U(F) \cup H' \tag{3}$$
$$S(F') = H' \backslash U(F) \tag{4}$$

That is, the resulting universe contains the input universe and all derived graphs, and the resulting subset contains all new graphs which was not known before. The removal of known graphs from the output subset is motivated by the goal of exploring the underlying network of derivations.

With the definition above we rewrite our initial example as; find $F_2 = p(F_1)$ for $F_1 = p(F)$ and $S(F) = U(F) = \mathfrak{U}$. The application $p(F_1)$ can now only discover derivations with at least one graph from $S(F_1)$, which by definition contains only new graphs. Therefore, only new derivations are found. Fig. 4 contains a visualization of the example.

The implementation utilizes the algorithm for transformation by first partially applying the rule to the subset of the input state, and then afterwards the full universe.
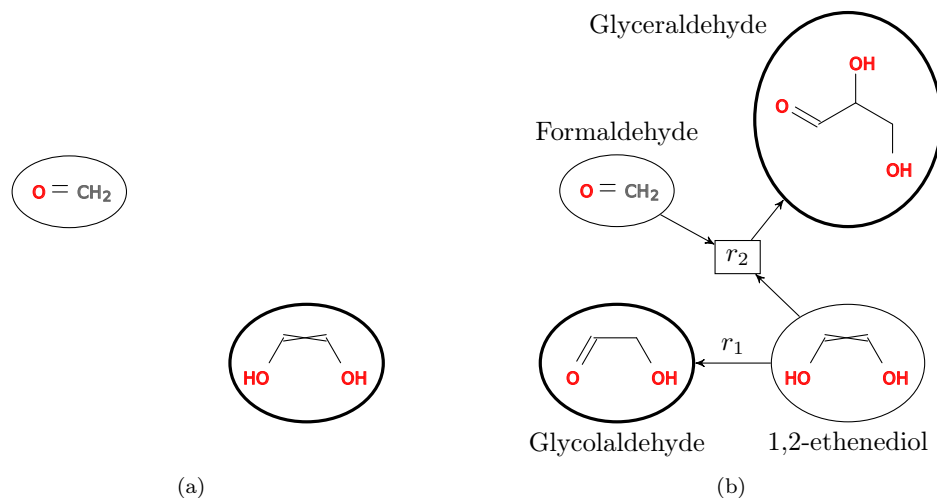
6

Figure 5: Application of a parallel strategy $Q = \texttt{parallel}[\{r_1, r_2\}]$ to a graph state F, with r_1 being the transformation rule for the enol to keto conversion and r_2 being the transformation rule for aldol addition (see Appendix A). (a) the reaction network with the graph state F consisting of $U(F) = \{\text{formaldehyde}, 1,2\text{-ethenediol}\}$ and $S(F) = \{1,2\text{-ethenediol}\}$. (b) the reaction network after evaluation of $Q(F)$, with two new molecules; glycolaldehyde and glyceraldehyde. The resulting graph state F' has $U(F') = \{\text{formaldehyde}, 1,2\text{-ethenediol}, \text{glycolaldehyde}, \text{glyceraldehyde}\}$ and $S(F') = \{\text{glycolaldehyde}, \text{glyceraldehyde}\}$. In both networks the subset of the graph state is highlighted.

Strategies

The previous section described how a rule p is applied to a state F to calculate a new state F', and motivated this by the example of composition of rule application, $F' = p(p(F))$. Using the definition of a graph state, we generalize the interface for rule application into general strategies. A strategy is simply any function Q from and to the set of graph states.

In the following we introduce core strategies defined in the framework. Most of the strategies are parameterized, which we will note with brackets around these parameters. The application of a strategy Q with some fixed parameter, n, to a graph state F is thus denoted as $Q[n](F)$.

Parallel

A parallel strategy is defined in terms of a set of substrategies, $\{Q_1, Q_2, \ldots, Q_n\}$. The result of applying a parallel strategy is the union of the results from applying the individual substrategies:

$$F' = \texttt{parallel}[\{Q_1, Q_2, \ldots, Q_n\}](F)$$
$$U(F') = \bigcup_{1 \leq i \leq n} U(Q_i(F))$$
$$S(F') = \bigcup_{1 \leq i \leq n} S(Q_i(F))$$

A simple use of parallel strategies is to model the possibility of different reaction mechanisms happening simultaneously. As example, consider modeling the formose chemistry which consists of keto-enol tautomerism and aldol addition, both reversible reactions (see Appendix A for the grammar details). Let r_1 and r_2 denote the corresponding reactions from Appendix A, i.e., the enol-to-keto reaction pattern for the carbonyl group and the pattern for aldol addition. The parallel strategy $Q = \texttt{parallel}[\{r_1, r_2\}]$ thus models that these two reactions can happen simultaneously as illustrated in Fig. 5.
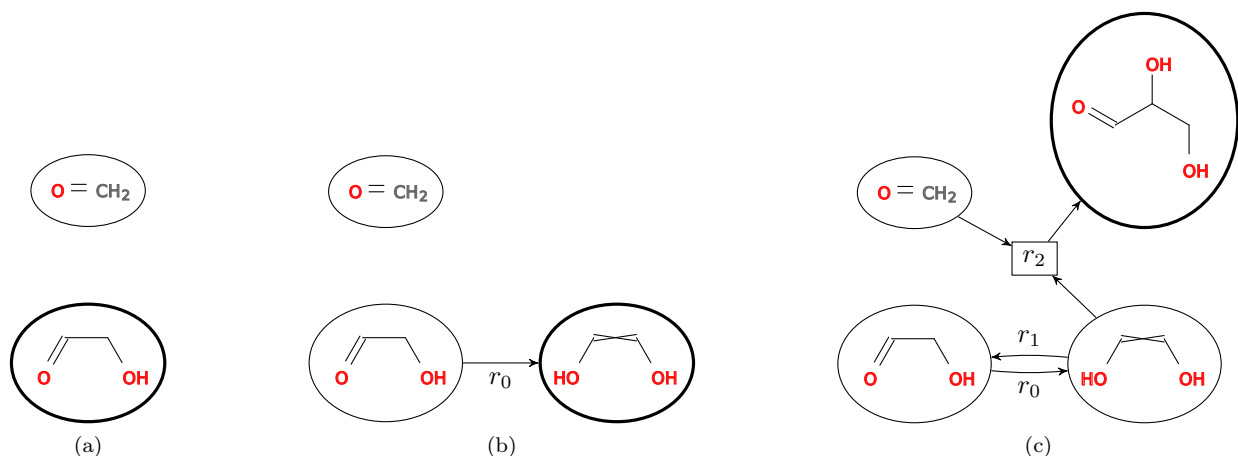
7

Figure 6: Application of the sequence strategy $Q = \mathtt{parallel}[\{r_0, r_1, r_2, r_3\}] \rightarrow \mathtt{parallel}[\{r_0, r_1, r_2, r_3\}]$ to the graph state F_0, with r_i denoting the transformation rules for keto-enol tautomerism and reversible aldol addition. (a) the initial reaction network with F_0 in which $U(F_0) = \{\text{formaldehyde, glycolaldehyde}\}$ and $S(F_0) = \{\text{glycolaldehyde}\}$. (b) the intermediary reaction network after evaluation of the first step of the strategy. The difference in graph state is that 1,2-ethenediol is now added to the universe and subset, while glycolaldehyde no longer is in the subset. (c) the reaction network after complete evaluation of $Q(F_0)$. The final graph state F_2 has all four molecules in the universe and only glyceraldehyde in the subset. Note that in the last step of the strategy the reverse keto-enol reaction is discovered, but glycolaldehyde is already in the universe so it will not be added to the subset of F_2. The subset of the graph state is highlighted in each network.

Sequence

A sequence strategy, Q, is a composition of a list of substrategies, $Q_1, Q_2, \ldots, Q_n$:

$$Q(F) = Q_n(\ldots (Q_2(Q_1(F))))$$

To increase left-to-right readability of sequence strategies, we will use the notation $Q = Q_1 \rightarrow Q_2 \rightarrow \cdots \rightarrow Q_n$. Additionally, if $Q_1 = Q_2 = \cdots = Q_n = Q'$, we may use the normal notation for powers of functions, $Q = Q'^n$, for the sequence.

An example of the application of a sequence strategy can be seen in Fig. 6, in which two sequential steps of the formose chemistry (parallel strategies) are derived starting from a graph state F with $U(F) = \{\text{formaldehyde, glycolaldehyde}\}$ and $S(F) = \{\text{glycolaldehyde}\}$.

Repetition

The sequencing strategy only allows composition of a fixed number of strategies, whereas the repetition strategy is used to compose a single strategy with itself many times.

A repetition strategy, Q, is parameterized by a non-negative integer, n, and an inner strategy Q'. The inner strategy is composed with itself until the graph state reaches a fixed point or its subset is empty, however at most n times:

$$Q = \mathtt{repeat}[Q', n] = Q'^k$$
$$k = \min\{0, 1, \ldots, n\}, \text{ such that } Q'^k(F) = Q'^{k+1}(F) \vee S(Q'^{k+1}(F)) = \emptyset \vee k = n$$

This means that if the graph state reaches a fixed point then that graph state is returned, and if the subset of the state becomes empty then the *previous* state is returned. We motivate this condition of a non-empty subset of a produced graph state by our definition of rule application, which requires at least one graph from the subset. By returning the last graph state with non-empty subset the repetition strategy can be used as a precomputation in a sequence to find a kind of closure under some inner strategy.
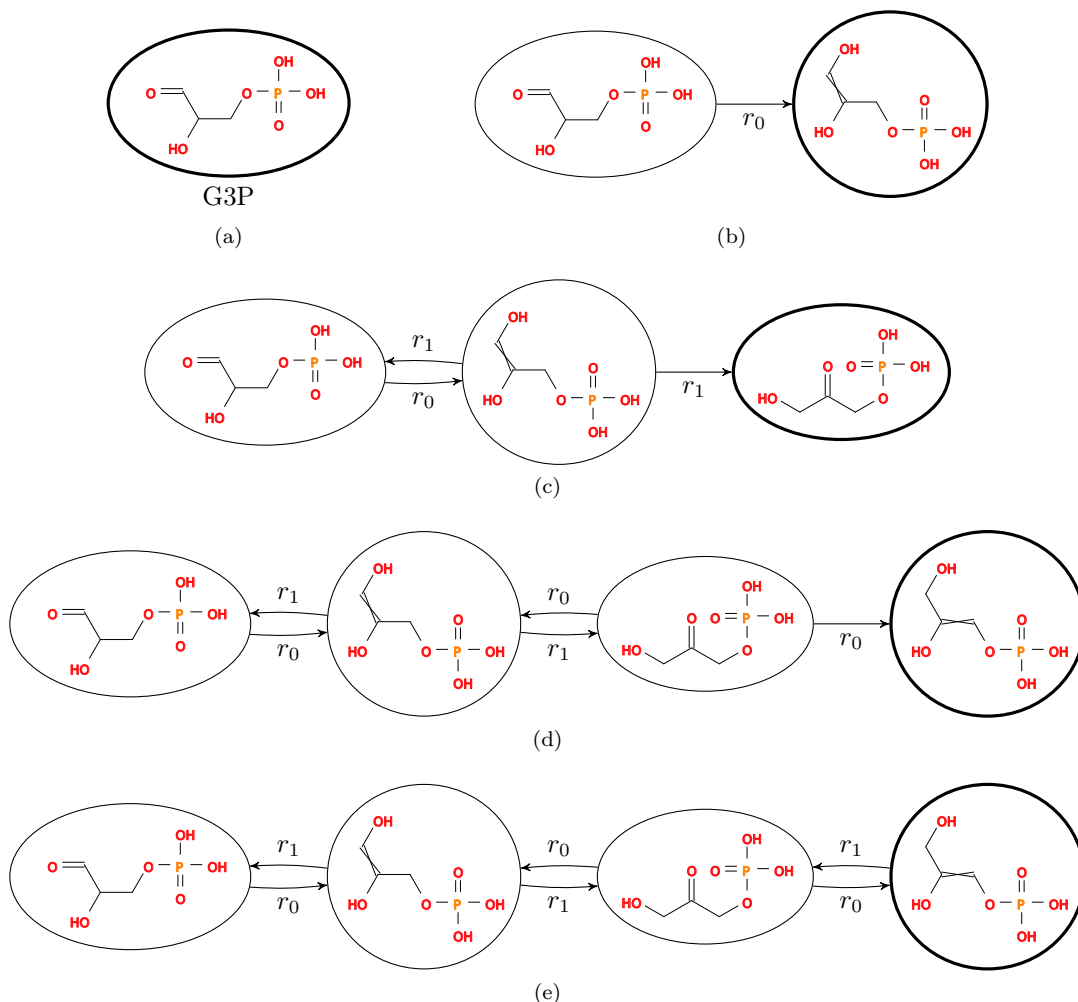
8

Figure 7: The strategy $Q = \texttt{repeat}[\texttt{parallel}[\{r_0, r_1\}]]$ applied to the the initial graph state F_0 with $U(F_0) = S(F_0) = \{\text{G3P}\}$ (shown in (a)). (b)–(d) the intermediary reaction networks from evaluation of $Q(F_0)$. Each step discovers a new isomer which constitutes the new subset. Additionally, the reaction to the previous isomer is discovered. However, this molecule is already in the universe of the current graph state and is therefore not added to the subset. (e) the final step in the repetition results in an empty subset as only known molecules (those in the universe) are rediscovered. The graph state from (d) is therefore the result. In all networks the subset of the current graph state is highlighted.

Note that if $k = 0$ the strategy becomes the identity strategy, i.e., the resulting graph state is the same as the input graph state. If n is set large enough to not limit the repetition, we call it unbounded repetition, and write it as $Q = \texttt{repeat}[Q']$.

In Fig. 6 the strategy for deriving two steps of the formose network is shown. As a generalization the strategy $Q = \texttt{repeat}[n, \texttt{parallel}[\{r_0, r_1\}]]$ can be used to derive (at most) n steps of the network. Fig. 7 shows another example using the repetition strategy, where all isomers of glyceraldehyde 3-phosphate (G3P) are generated.

Revive

Consider the following high-level description of a strategy: Given a single graph g, try to apply the rule p. If the application of p is successful, then let H denote all the produced graphs and return $H \backslash \{g\}$ (all graphs
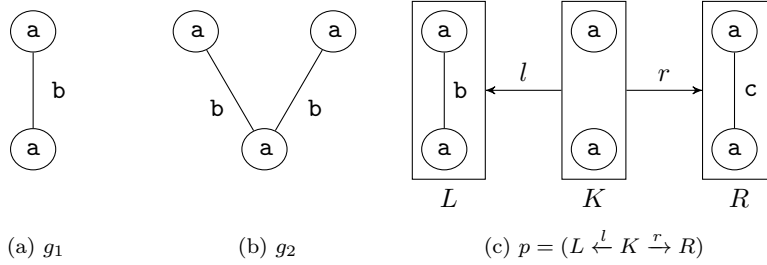
9

Figure 8: Graphs and transformation rule for the example of the semantics of revive strategies.

not already known). If the application of p is not successful, then intentionally $\{g\}$ should be returned. The simple strategy $Q = p$ applied to F with $S(F) = U(F) = \{g\}$ only partially achieves this, as illustrated in the following. Let $F' = Q(F)$ be the resulting graph state after evaluation of the strategy on F. Using the definition of the rule application strategy, Eq. (2)–(4), we get

- $S(F') = H \backslash \{g\}$ and $U(F') = H \cup \{g\}$ if p is successfully applied, and

- $S(F') = \emptyset$ and $U(F') = \{g\}$ if p can not be applied.

However, the desire was to have $S(F') = \{g\}$ in the unsuccessful case. The intention of the revive strategy is to provide a mechanism to model the desired behaviour. A rule application strategy discovers a (possibly empty) set of derivations. We say that a graph g is *consumed* in a rule application strategy if any of the discovered derivations $G \Rightarrow H$ have $g \in G$. In the natural way we extend this and say that a graph g is consumed by a strategy if it is consumed by any of its substrategies. A revive strategy, $\texttt{revive}[Q']$, is parameterized by a single substrategy, Q', and is defined as:

$$F' = \texttt{revive}[Q'](F)$$
$$U(F') = U(Q'(F))$$
$$S(F') = S(Q'(F)) \cup \{g \in S(F) \mid g \in U(F') \wedge g \text{ is not consumed in } Q'\}$$

That is, any graph from the input subset which is still in the output universe and was not consumed, will be added to the output subset. The high-level described example to illustrate the problem with a simple rule p can now be solved with the strategy $Q = \texttt{revive}[p]$. If the application of p is unsuccessful, then g is not consumed and will be added to the resulting subset.

As another example, consider the following problem. Two graphs, g_1 and g_2 and the transformation rule p, as illustrated in Fig. 8 are given. We wish to develop a strategy to transform all edge labels using rule p, with the intend to use this strategy as a precomputation for a subsequent strategy. That is, the subset of the graph state after evaluation of the strategy must contain the completely transformed graphs in the subset. The strategy $Q = \texttt{repeat}[p]$ may seem like the most intuitive approach to model this process. However, the evaluation of $Q(F)$ with $S(F) = U(F) = \{g_1, g_2\}$ does not give the intended result, which is illustrated in Fig. 9. The problem is that the repetition strategy will continue as long as *any* new graph can be discovered, and does not preserve the most derived graphs in the subset. Using the strategy $\texttt{repeat}[\texttt{revive}[p]]$ correctly solves the problem. A chemical example for the revive strategy will be given in the results section.

Derivation Predicates

For the purpose of precise modeling and the problems with combinatorial explosion it is convenient to limit the possibilities of expansion. We define two variations of the concept of derivation predicates, which both introduce extra constraints in Eq. (2) to prune unwanted derivations. The strategy $\texttt{leftPredicate}[P, Q']$ is defined by the predicate P on a multiset of graphs and a transformation rule, and by the substrategy Q'. A
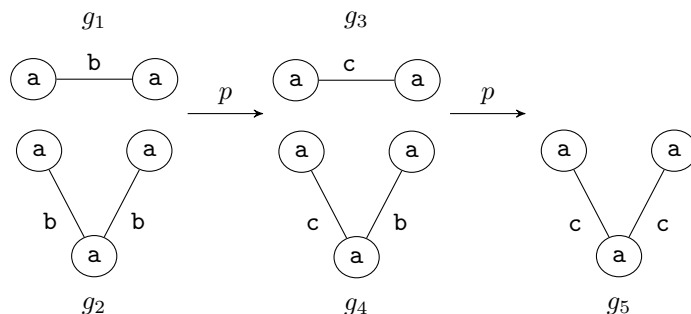
Figure 9: Illustration of the application of `repeat[p]` to F with $S(F) = U(F) = \{g_1, g_2\}$. Only the subset of the graph states are shown. The first application of p results in two new graphs, g_3 and g_4, but as p can only be applied to g_4 the final subset is only a single graph, g_5, instead of both g_3 and g_5.

candidate derivation from the graphs G with the rule p found by Q', is only fully calculated and accepted if $P(G, p)$ is true. A right predicate strategy, `rightPredicate[P, Q']` is also defined by a predicate and a substrategy, though with the predicate P evaluating a complete derivation. Thus, a derivation $G \overset{p}{\Rightarrow} H$ is only accepted if $P(G \overset{p}{\Rightarrow} H)$ is true.

As example, given a strategy Q' we wish to produce only graphs with at most 42 vertices (atoms, in a chemical context). This requires a right predicate strategy as information about the right side of the derivation (the products) are needed. This can be specified with the following strategy:

$$Q = \texttt{rightPredicate}[P, Q']$$

$$P(G \overset{p}{\Rightarrow} H) \equiv \forall h \in H : |V(h)| \le 42$$

Instead, we might want to restrict that some molecule g should not be an educt in any reaction with the transformation rule being r. This constraint does not require the information of a complete derivation, and may as such be formulated as a left predicate strategy:

$$Q = \texttt{leftPredicate}[P, Q']$$

$$P(G, p) \equiv \neg(r = p \wedge g \in G)$$

with Q' being an arbitrary strategy.

Filter, Sort, Take and Add

To facilitate more elaborate use of strategies in a functional style we define several strategies which correspond to functions on lists in other languages. As a graph state is composed of both a universe and a subset, all of these strategies are defined in two variations.

A filter strategy is parameterized by a predicate on a graph and a graph state:

$$F' = \texttt{filterSubset}[P](F) \qquad\qquad F' = \texttt{filterUniverse}[P](F)$$
$$U(F') = U(F) \qquad\qquad U(F') = \{g \in U(F) \mid P(g, F)\}$$
$$S(F') = \{g \in S(F) \mid P(g, F)\} \qquad\qquad S(F') = \{g \in S(F) \mid P(g, F)\}$$

A sorting strategy is parameterized with a predicate on two graphs and a graph state, used as a less-than predicate in a stable sort of a list of graphs:

11

$$F' = \texttt{sortSubset}[P](F) \qquad\qquad F' = \texttt{sortUniverse}[P](F)$$
$$U(F') = U(F) \qquad\qquad U(F') = \texttt{stableSort}[P](U(F))$$
$$S(F') = \texttt{stableSort}[P](S(F)) \qquad\qquad S(F') = S(F)$$

The choice that the sorting algorithm must be stable is motivated by the desire to allow lexicographical sorting by sequencing several sorting strategies.

A take strategy is parameterized with a natural number:

$$F' = \texttt{takeSubset}[n](F) \qquad\qquad F' = \texttt{takeUniverse}[n](F)$$
$$k = \min\{n, |S(F)|\} \qquad\qquad k = \min\{n, |U(F)|\}$$
$$U(F') = U(F) \qquad\qquad U(F') = \{U(F)_1, U(F)_2, \ldots, U(F)_k\}$$
$$S(F') = \{S(F)_1, S(F)_2, \ldots, S(F)_k\} \qquad\qquad S(F') = S(F) \cap U(F')$$

An addition strategy appends a given set of graphs to either the universe and optionally also to the subset:

$$F' = \texttt{addSubset}[\{g_1, g_2, \ldots, g_n\}](F) \qquad\qquad F' = \texttt{addUniverse}[\{g_1, g_2, \ldots, g_n\}](F)$$
$$U(F') = U(F) \cup \{g_1, g_2, \ldots, g_n\} \qquad\qquad U(F') = U(F) \cup \{g_1, g_2, \ldots, g_n\}$$
$$S(F') = S(F) \cup \{g_1, g_2, \ldots, g_n\} \qquad\qquad S(F') = S(F)$$

An example usage of these strategies is the procedure of ranking graphs according to some property, take the best n graphs for subsequence expansion, i.e:

$$Q' = \texttt{sortSubset}[P] \to \texttt{takeSubset}[n]$$

Note that the sorting predicate P can be based on any external data such as results from wet lab experiments. As example we have used mass spectrometry data to bias the expansion towards high intensity molecules (see the Results and Discussion section).

The addition strategies can be used both for injecting new graphs in the middle of a strategy, but we also find them convenient simply for uniform left-to-right writing of a strategy application. E.g., given a (large) strategy Q we wish to apply to the graph state F, we can write:

$$F' := \texttt{addUniverse}[U(F)] \to \texttt{addSubset}[S(F)] \to Q$$

with the interpretation $F' = Q(F)$.

Implementation Remarks

The strategies are implemented in C++ as part of a library, to allow easy extension at the user level. Extensions can vary from simple graph state manipulating strategies to complete replacement of the underlying transformation formalism. The library is aimed at chemical graph transformation, with special optimization for molecules (e.g., use of canonical SMILES strings for graph isomorphism [20, 21]), but is not restricted to the domain of chemistry. The current implementation uses VF2 [22] to find subgraph isomorphisms, and as a fall-back algorithm for isomorphism check for general graphs. Furthermore, the library utilizes data structures and procedures for molecule handling form the Graph Grammar Library (GGL) [23]. A Python module with bindings to the C++ library is also implemented to allow easy development of expansion strategies.
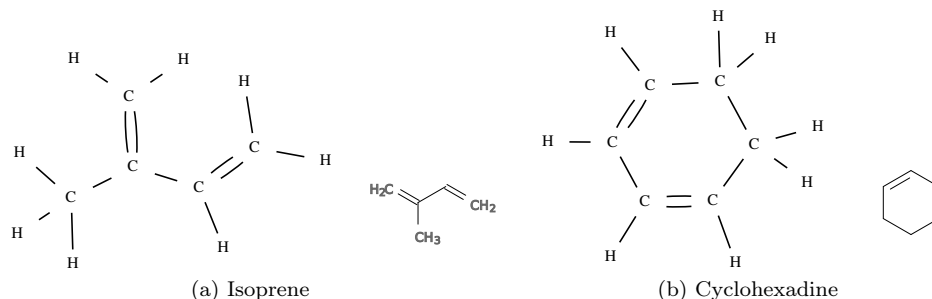
(a) Isoprene (b) Cyclohexadine

Figure 10: The starting molecules, (a) isoprene and (b) cyclohexadine, for application of the Diels-Alder reaction. The molecules are shown in two versions; one with all vertices explicit and chemical interpretation of edge labels (left), and one version in standard chemical visualization.

Results

In this section we will apply our strategy framework to three different chemical systems and present results on how to systematically explore complex chemical universes: i.) for the Diels-Alder reaction system we will repeatedly merge molecules with isoprene, ii.) we will compare chemical universes of basic formose chemistry with and without using borate as inhibitor motivated by a recent experiment by [24], and iii.) we will present a strategy to explore the complex chemical spaces of hydrogen cyanide polymerization and hydrolysis product in order to show how to integrate mass spectrometry results in our framework. In order to easily illustrate subspaces that are also expected to exist in a chemical setting, we will apply the strategy framework to a small puzzle game (Appendix C).

The Diels-Alder Reaction

The Diels-Alder reaction is one of the most useful reaction in organic chemistry and has heavily influenced total synthesis in the last decades [25]. The explosion of the chemical space by applying this reaction several times will be biased by the strategy framework. The reaction is shown in an example derivation in Fig. 1, while the starting molecules, isoprene and cyclohexadine, are shown in Fig. 10. Let $p = (L \leftarrow K \rightarrow R)$ be the transformation rule modeling the Diels-Alder reaction. The intention of the rule is that it is applied to two molecules, but this constraint is not encoded in the rule. We therefore first wrap p with a derivation predicate:

$$Q_p = \texttt{leftPredicate}[P, p] \qquad\qquad P(G, p') \equiv \#G = 2$$

This means that all derivations $G \overset{p}{\Rightarrow} H$ must have $|G| = 2$.

A generic breadth-first exploration of the chemical space can be done with the following strategy:

$$Q_{\text{BFS}} = \texttt{addSubset}[\{\text{isoprene}, \text{cyclohexadine}\}] \rightarrow \texttt{repeat}[Q_p, n]$$

However, for $n = 4$ the strategy already discovers 825 new graphs through 1278 derivations.[1] The number of subgraph isomorphism queries throughout the evaluation is 74591. In Appendix B, Fig. 14 the resulting derivation graph for just $n = 2$ is shown.

We now decide to only look at the subspace of molecules which are derived by repeatedly merging molecules with isoprene, starting with cyclohexadine. The following strategy implements this specification:

$$
\begin{aligned}
Q_{\text{subspace}} = {}& \texttt{addUniverse}[\{\text{isoprene}\}] \rightarrow \texttt{addSubset}[\{\text{cyclohexadine}\}] \qquad\qquad (5) \\
& \rightarrow \texttt{leftPredicate}[P_{\text{init}}, Q_p] \rightarrow \texttt{filterUniverse}[P_{\text{filter}}] \\
& \rightarrow \texttt{repeat}[Q_p, n]
\end{aligned}
$$

---

[1]In this scenario we regard derivations which only differ in the matching morphism as duplicates. The evaluation of the strategy takes in the order of 10 seconds with a Intel® Core™ i5-2500K CPU (3.30GHz).
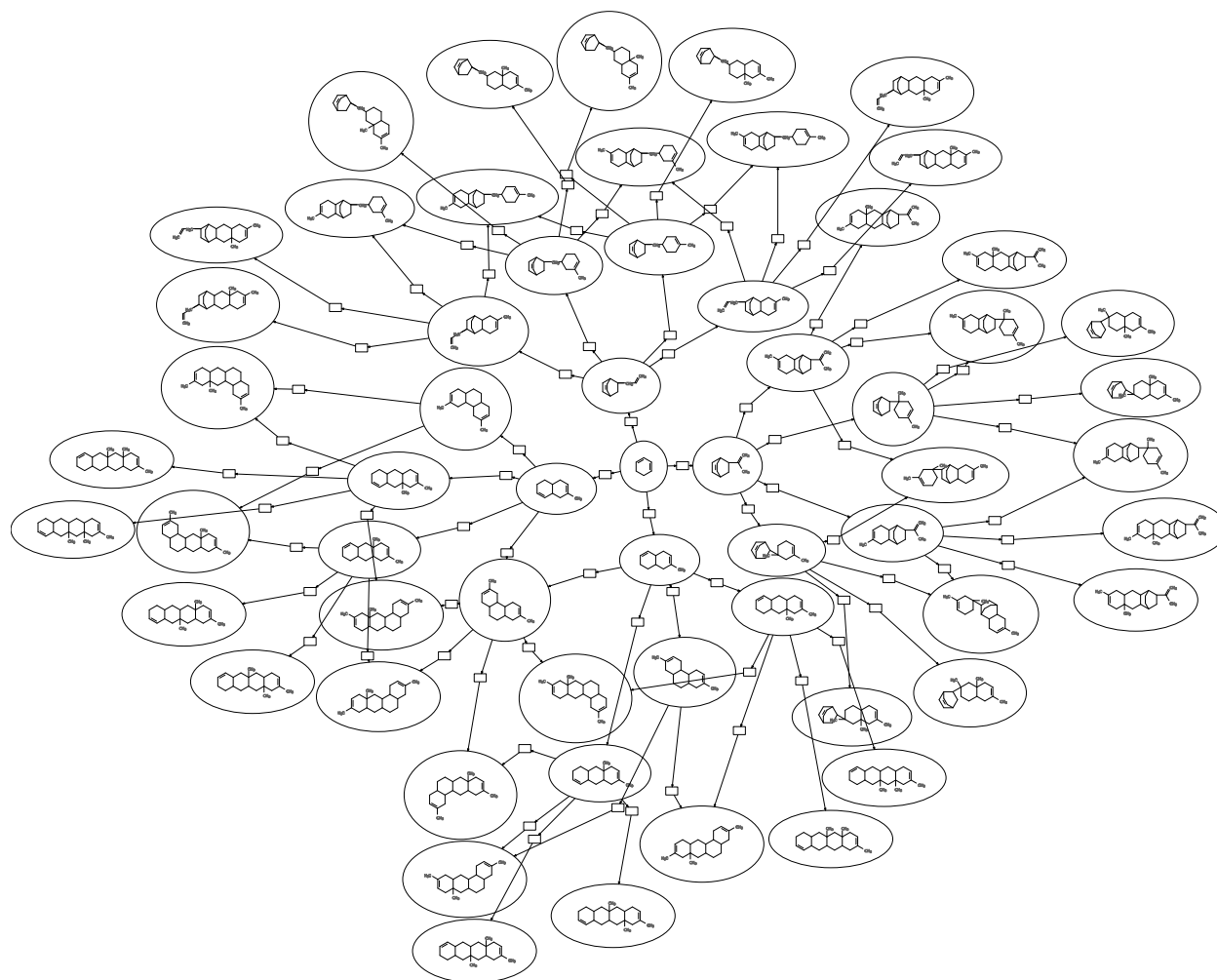
13

Figure 11: The derivation graph resulting from evaluating the expansion strategy Q_{subspace}, Eq. (5). To minimize clutter, the vertex with isoprene and the corresponding edges are not shown, although isoprene is involved in any reaction (the resulting chemical reaction network is a hypergraph).

with

$$P_{\text{init}}(G, p') \equiv G = \{\text{isoprene}, \text{cyclohexadine}\}$$
$$P_{\text{filter}}(g, F) \equiv g \neq \text{cyclohexadine}$$

This first computes all possible proper derivations $\{\text{isoprene}, \text{cyclohexadine}\} \overset{p}{\Rightarrow} H$, then removes cyclohexadine from the graph state to prevent further derivations. In the end it uses breadth-first expansion for at most n steps. This strategy, with $n = 3$ (i.e., 4 expansion steps including the very specific first step) discovers only 165 new graphs through 236 derivations,[2] and uses 5524 subgraph isomorphism queries. The derivation graph with $n = 2$ is visualized in Fig. 11.

---

[2]In this scenario we regard derivations which only differ in the matching morphism as duplicates. The evaluation of the strategy takes in the order of 8 seconds with a Intel® Core™ i5-2500K CPU (3.30GHz).

Borate stabilized Formose Reaction

Sugars, or more general carbohydrates, a broad class of organic compounds, can be viewed as polymers of formaldehyde units. The reactivity of carbohydrates is dominated by their carbonyl and their vicinal alcohol functional groups. In particular the enolized form of a carbonyl group may attack another one (in keto form), resulting in the formation of a new carbon-carbon bond. This reaction is known as *aldol addition* (see Fig. 6c). If the carbon atom adjacent to a carbonyl group carries an alcohol functionality, than the *enolization reaction* of the carbonyl group erases the "information" at which carbon atom the carbonyl functionality was located before the enolization. This effect allows the carbonyl group to "travel" along the carbohydrate backbone (see Fig. 7e). Both reactions are responsible for the meta-stability of carbohydrates and result in complex carbohydrate mixtures when repeated again and again as for instance under the conditions of the formose reaction [26]. The formose reaction has been extensively discussed as a possible prebiotic route to higher carbohydrates in particular five-carbon sugars, such as ribose, needed for the formation of nucleotides (the building blocks of RNA) [27]. Unfortunately, if the formose reaction is not stopped in time the reaction mixture turns into black "tar". Therefore, some stabilizing mechanism compatible with prebiotic environments, that prevent the destruction of interesting sugars, is indispensable to keep the formose reaction as a plausible prebiotic scenario for higher carbohydrate formation. The addition of borate, capable of binding vicinal diols, to the reaction mixture has been identified as such a stabilizing mechanism, that biases the outcome of the formose reaction towards high yields of five-carbon sugars [24]. In the following we illustrate how expansion strategies can be exploited to carve out the differences between the formose reaction networks with and without borate.

The basic formose reaction consists of two types of reversible reaction patterns, keto-enol tautomerism and aldol reaction. As they are reversible they are modeled by two transformation rules each. These are shown in Appendix A as transformation rule $r_0, \ldots, r_3$, while the two initial molecules, formaldehyde and glycolaldehyde, are shown in Fig. 12a and 12b respectively. To keep the model simple we use a borate-like molecule, Fig. 12c, with just two hydroxyl groups instead of a complete molecule. To enable the formation of borate complexes we use the transformation rule shown in Fig. 12d. This reaction pattern is described in [28] as inhibiting keto-enol tatutomerism by making the hydrogen atoms attached to the carbon atoms non-acidic. To approximate this behaviour we relabel these vertices from H to D, thereby preventing the reaction pattern of enolization (r_0 in Appendix A) from matching at these locations. The relabeling is done with the reaction 'hToD', Fig. 12e.

The formose chemistry contains an infinite number of molecules, so to limit the scope of the exploration we prune any reaction which creates molecules with more than 5 carbon atoms. This is formulated with a right predicate strategy around the application of the basic formose reaction patterns:

$$\texttt{rightPredicate}[P_{\#C}, \texttt{parallel}[\{r_0, r_1, r_2, r_3\}]]$$

$$P_{\#C}(G \overset{p}{\Rightarrow} H) \equiv \forall h \in H : h \text{ has at most 5 carbon atoms}$$

As a reference, we generate the non-inhibited reaction network with the strategy Q_{BFS}:

$$
\begin{aligned}
Q_{\text{BFS}} = &\ \texttt{addUniverse}[\{\text{formaldehyde}\}] \\
&\rightarrow \texttt{addSubset}[\{\text{glycolaldehyde}\}] \\
&\rightarrow \texttt{repeat}[\\
&\qquad \texttt{rightPredicate}[P_{\#C}, \texttt{parallel}[\{r_0, r_1, r_2, r_3\}]] \\
&\ ]
\end{aligned}
$$

Not all molecules can actually bind with borate and must therefore be preserved while the other molecules form complexes. This is modeled with a revive strategy around the actual complex forming reaction pattern, 'addBorate'. After the potential forming of a borate complex, the relevant hydrogen atoms must be made inactive using the rule 'hToD'. The number of relevant hydrogens may not be the same for alle molecule and therefore the relabeling strategy is embedded in both a repeat and revive strategy. This models the notion

(a) Formaldehyde

(b) Glycolaldehyde

(c) Borate

(d) Borate + 1,2-diol reaction pattern, 'addBorate'

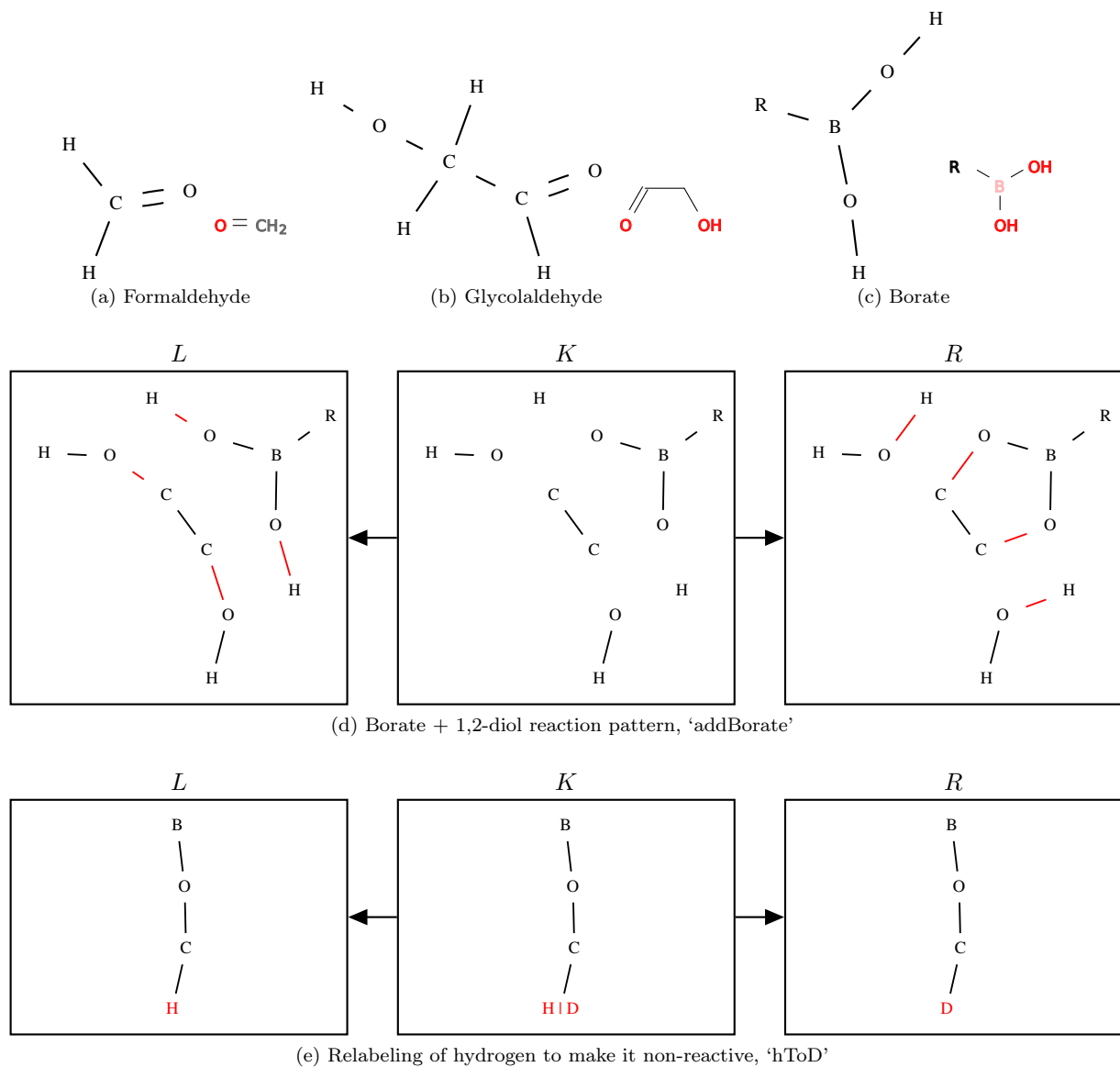(e) Relabeling of hydrogen to make it non-reactive, 'hToD'

Figure 12: (a)–(c) the starting molecules of the borate inhibited formose reaction. The three molecules are shown both as explicit graphs with all vertices and in standard chemical visualization. The borate molecule (c) is modeled with only two hydroxyl groups to simplify the model. (d) the reaction pattern for forming borate complexes with 1,2-diols. This rule additionally has a matching constraint: none of the carbon atoms may be an endpoint of a double bond. To approximate the subsequent non-reactivity of the hydrogens on the carbon atoms we relabel them to D using the reaction 'hToD' (e). This relabeling is in the context graph, K, represented with the annotation H | D.

of "as many times as possible" on a collection of molecules. The reaction network with borate inhibition can

thus be calculated by the following strategy:

$$Q_{\text{borate}} = \texttt{addUniverse}[\{\text{formaldehyde}, \text{borate}\}]$$
$$\rightarrow \texttt{addSubset}[\{\text{glycolaldehyde}\}]$$
$$\rightarrow \texttt{repeat}[$$
$$\texttt{revive}[\text{addBorate}]$$
$$\rightarrow \texttt{repeat}[\texttt{revive}[\text{hToD}]]$$
$$\rightarrow \texttt{rightPredicate}[P_{\#C}, \texttt{parallel}[\{r_0, r_1, r_2, r_3\}]]$$
$$]$$

Let $\mathcal{G}$ denote the set of molecules used and generated by the evaluation of Q_{borate} on the empty graph state. This set of molecules contain both borate complexes and simple carbohydrates without boron. To canonicalize the molecules we can use the strategy

$$Q_{\text{canon}} = \texttt{addSubset}[\mathcal{G}] \rightarrow \texttt{repeat}[\texttt{revive}[\text{dToH}]] \rightarrow \texttt{repeat}[\texttt{revive}[\text{removeBorate}]]$$

with 'removeBorate' being the inverse transformation rule of 'addBorate', and 'dToH' being the inverse of 'hToD'. Note that 'removeBorate' requires water molecules as educts, but if 'addBorate' was ever used in Q_{borate} these molecules must be in $\mathcal{G}$.

As a variant of the network, we also calculate the network with a an extra molecule, dihydroxyacetone, in the subset:

$$Q_{\text{borate}}^+ = \texttt{addUniverse}[\{\text{formaldehyde}, \text{borate}\}]$$
$$\rightarrow \texttt{addSubset}[\{\text{glycolaldehyde}, \text{dihydroxyacetone}\}]$$
$$\rightarrow \texttt{repeat}[$$
$$\texttt{revive}[\text{addBorate}]$$
$$\rightarrow \texttt{repeat}[\texttt{revive}[\text{hToD}]]$$
$$\rightarrow \texttt{rightPredicate}[P_{\#C}, \texttt{parallel}[\{r_0, r_1, r_2, r_3\}]]$$
$$]$$

In Fig. 13 the reference reaction network created with Q_{BFS} is shown. Reactions in black are active only in the basic formose reaction case with formaldehyde and glycolaldehyde as set of input molecules. If borate is added to the input set of molecules, the reactions highlighted in blue are active, while the rest of the network is inactive. Finally if dihydroxyacetone is added to the input set of molecules the reactions highlighted in green are activated in addition to the blue part of the network. The evaluation of Q_{borate} leaves only the blue reactions, which are selective pathways from glycolaldehyde (C2a) to five-carbon sugars (C5b, C5l$_1$, C5l$_2$) active, while the rest of the network is shut down via borate inhibition. These pathways rely on a constant replenishment of glycolaldehyde. Here dihydroxyacetone (C3k) comes into play. C3k can only be formed from within the formose network via retro-aldol reaction from higher carbohydrates. If added to the reaction network an catalytic loop is activated (sub-network in green: C3k, C3e, C4k, C4e, C5b, retro-aldol red dashed arrow to C3e and C2a) supporting the blue sub-network since C2a ends up as some five-carbon sugars in the blue sub-network. C3e enters another round in the cycle to construct another C2a. These computational results are in very good agreement with the experimental results presented in [30].

HCN Polymerization and Hydrolysis biased by Mass Spectrometry Results

Hydrogen cyanide (HCN) is a known prebiotic precursor of amino acids as well as many other molecules relevant to present-day biology. It has been used to synthesize adenine already in 1961 [31] amino acids [32], as well as many other molecules relevant to present-day biology [32–38], and it is also known to play a key role
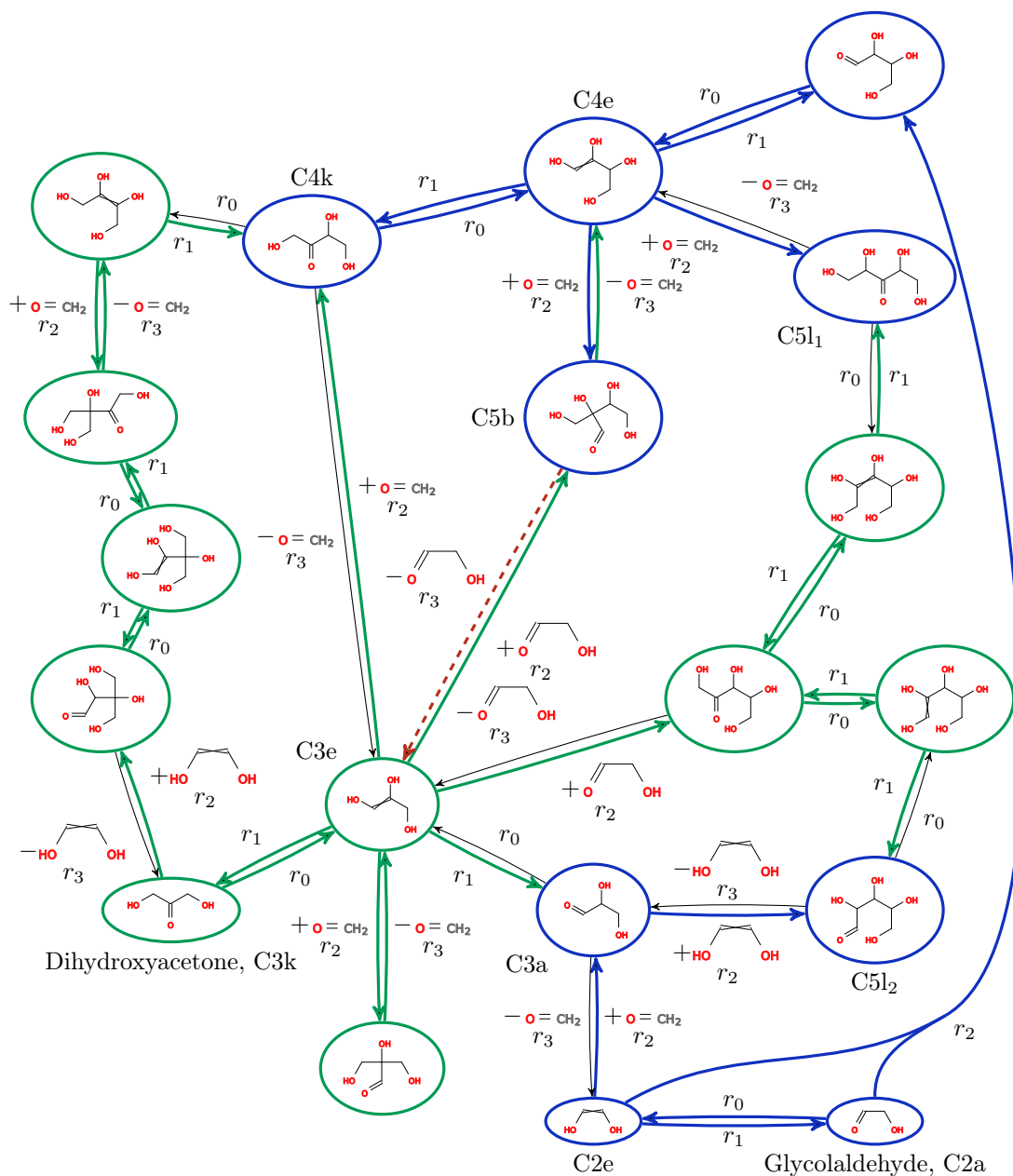
Figure 13: The reaction network of the formose chemistry as calculated with the strategy Q_{BFS}. The blue subnetwork correspond to the borate inhibited network calculated with Q_{borate}. The green and blue networks together with the red reaction (C5b to C3e) correspond the network calculated with Q_{borate}^{+}, i.e., with dihydroxyacetone as an input compound. Note that this particular model does not include stereochemical properties, and that the molecule depictions are made using Open Babel [29], which for instance means crossing double are used to indicate the unspecified stereo. Each reaction is annotated with the reaction pattern, r_i, used to realize the concrete reaction. For the aldol reactions, r_2 and r_3, the secondary educt $(+)$ or product $(-)$ is additionally shown. The addition of borate in Q_{borate}^{+} is done with the strategy $\texttt{revive}[addBorate]$, meaning that at most 1 borate is added in each iteration. The red reaction is no longer available if the addition is done with the strategy $\texttt{repeat}[\texttt{revive}[addBorate]]$, meaning "add as many as possible".

also in sugar synthesis [39]. In [40] graph grammar approaches and mass spectrometry results were integrated in order to generate a chemical network with highly likely polymerization/hydrolysis products. In the first step of the wetlab experiments acid-catalyzed HCN polymers were created, in the second step the polymers were hydrolysed under different conditions. The mass spectrometry results of the wetlab experiments were used in order to bias the chemical space exploration performed with the strategy framework. A detailed discussion of the results including a large variety of adenine pathways and autocatalytic processes within the inferred chemical space can be found in [40]. Here we focus on the description of the used strategies.

The model of the HCN chemistry is based on many transformation rules which are shown in detail in the web supplement of [40]. For the purpose of a concise strategy description we let $\mathfrak{R}$ denote the set of needed transformation rules. The expansion strategy is aimed at modeling the wetlab experiments and thus consist of the sequencing of a strategy for polymerization with a strategy for hydrolysis. As these two strategies are quite similar we only state the hydrolysis strategy, $Q_{\text{hydrolysis}}$.

Ideally, a simple breadth-first expansion strategy, $\texttt{repeat}[\mathfrak{R}]$, can be used to expand the network but due to the sheer combinatorial explosion only very few steps can be calculated. Instead the following strategy can be used to prune the expansion:

$$
\begin{aligned}
Q_{\text{hydrolysis}} = \texttt{addSubset}&[\{\text{HCN}, \text{NH}_4, \text{H}_2\text{O}, \text{OH}^-\}] \\
\rightarrow \texttt{repeat}\big[& \\
& \texttt{leftPredicate}[P, \texttt{parallel}[\mathfrak{R}]] \\
\rightarrow\, & \texttt{filterUniverse}[P_{\text{isomer}}] \\
\rightarrow\, & \texttt{sortUniverse}[P_{\text{intensity}}] \\
\rightarrow\, & \texttt{takeUniverse}[20] \\
\rightarrow\, & \texttt{addUniverse}[G_{\text{small}}] \\
\big]&
\end{aligned}
$$

where the predicates are defined as

$$P(G, p) \equiv \text{at most 1 molecule of } G \text{ has molar mass greater than 50}$$
$$P_{\text{isomer}}(g, F) \equiv true \text{ iff the normalized Boltzmann factor of } g \text{ is above a certain threshold.}$$
$$\text{The factor is calculated based on the isomers of } g \text{ in } U(F)$$
$$P_{\text{intensity}}(g_1, g_2, F) \equiv intensity(g_1) > intensity(g_2)$$
$$\text{The intensities are found in the mass spectrometry data using the molar masses}$$

That is, the input graph state is augmented with basic food molecules. Then the main hydrolysis step is repeated until no new molecules are found. The main step first expands the network under the constraint that at least one small molecule is an educt in each reaction, which limits the growth of the molecules to be linear as opposed to exponential. The subsequent three steps prunes the graph state of unlikely molecules, first by calculating normalized Boltzmann factors within each class of isomers. Then the mass spectrometry data from the wetlab experiments are used to select the 20 molecules with highest intensity for the next expansion step. These pruning steps might have removed the basic food molecules, and they are therefore reintroduced. Additionally the molecules immediately derivable from the food molecules are added. The evaluation of the overall HCN strategy take considerably longer (hours) to calculate than the previous examples. The bulk of the time is however spent on calculating energy value used the Boltzmann factors. For further details see [40].

Conclusions

We have introduced here a generic framework to specify and execute strategies for the systematic exploration of spaces of graphs. Our generative approaches use the Double Pushout formalism to derive new graphs. Since

this task is of immediate practical relevance in chemistry, we designed our framework and implementation with the aim of high efficiency in this particular domain of application. As performance was a particular focus of our work, we use state-of-the-art subgraph isomorphism check methods and we heavily employ hashing techniques in the checks for graph isomorphism; in order to infer proper derivations of new molecules with full or partial rule application we do *not* use a straightforward method to enumerate all possible left-hand-sides of derivations. Instead we employ partial rule applications, a method that shows theoretically as well as empirically a much better performance. The latter aspect will discussed in more detail elsewhere.

As showcase examples we have considered complex systems of chemical reactions. For Diels-Alder reactions, which is plagued by a very rapid combinatorial explosions, we used the strategy framework to guide the exploration to emphasize products of repeated isoprene addition instead of unconstrained combinations of reactions products. This is of relevance e.g. in terpene chemistry and biosynthesis. In the case of the formose reaction we show how the strategies framework can be applied to explaining the effects of additional reactants on a given reaction network. In particular, we can in rule based manner also determine which reactions are effectly superseeded by new ones, so that additional reactants can lead to a reduction of chemical network. The strategies framework thus serves not only as a convenient tool for exploration but allows also a detailed modelling of contraints in chemical networks.

Although the design was clearly chosen with systems chemistry and systems biology applications in mind, the strategy framework introduced here is however by no means limited to chemical applications. Another promising application is the emulation of higher-level rules. In the DPO graph grammar formalism, the size of a subgraph that is affected by a transformation is by construction bounded by the left graph of the production that is to be applied. Apparently simple operations on a graph, such as *"contract a clique in G to a single vertex"*, however, do not have such a bound since the clique sizes depend only on the input graph. Hence, such rules cannot be specified directly as productions in a DPO graph grammar. In the Appendix C we use the well-known Catalan game [41] to show how our strategy framework can be applied to emulate this type of higher-level rules.

In order to analyze chemical reaction networks as created by our strategy framework, there exist several mathematical techniques that we plan to apply to our generated networks. Two of the most prominent ones are Flux Balance Analysis [42] and Elementary Mode Analysis [43]. Note, that these methods are usually not applied to dynamically created reaction networks as produced by our framework. We aim at detecting new well-defined chemical reaction pattern. Furthermore, we expect to identify highly connected subgraphs in chemical spaces, that are connected via a small number of bridging reaction, similar to our observation for the Catalan game.

Authors contributions

J.L.A. implemented the strategy framework. All authors contributed to the theory, the writing of the manuscript and approved the submitted manuscript.

Acknowledgements

References

1. Eberhardt L, Kumar K, Waldmann H: **Exploring and exploiting biologically relevant chemical space**. *Curr Drug Targets* 2011, **12**:1531–1546.

2. Dow M, Fisher M, James T, Marchetti F, Nelson A: **Towards the systematic exploration of chemical space**. *Org. Biomol. Chem.* 2012, **10**:17–28.

3. Reymond JL, Awale M: **Exploring Chemical Space for Drug Discovery Using the Chemical Universe Database**. *ACS Chem. Neurosci.* 2012, **3**:649–657.

4. Wong YS: **Exploring Chemical Space: Recent Advances in Chemistry**. In *Chemical Genomics and Proteomics*, *Volume 800* 2012:11–23.

5. Ruddigkeit L, van Deursen R, Blum LC, Reymond JL: **Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17**. *Journal of Chemical Information and Modeling* 2012, **52**(11):2864–2875.

6. Dittrich P, Ziegler J, Banzhaf W: **Artificial chemistries - a review**. *Artificial life* 2001, **7**(3):225–275.

7. Fernández M, Namet O: **Strategic programming on graph rewriting systems**. In *Proceedings of the 1st International Workshop on Strategies in Rewriting, Proving, and Programming (IWS 2010)*, *Volume 44 of* Electronic Proceedings in Theoretical Computer Science 2010:1–20.

8. Fernández M, Kirchner H, Namet O: **A Strategy Language for Graph Rewriting**. In *Proceedings of the 21st International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2011)*, *Volume 7225 of* Lecture Notes in Computer Science 2012:173–188.

9. Andrei O, Fernández M, Kirchner H, Melançon G, Namet O, Pinaud B: **PORGY: Strategy driven interactive transformation of graphs**. In *In Proceedings of the 6th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2011)*, *Volume 48 of* Electronic Proceedings in Theoretical Computer Science 2011:54–68.

10. Pinaud B, Melançon G, Dubois J: **PORGY: A Visual Graph Rewriting Environment for Complex Systems**. *Comput. Graph. Forum* 2012, **31**(3).

11. Diels OPH, Alder K: **Synthesen in der hydroaromatischen Reihe**. *Justus Liebig's Annalen der Chemie* 1928, **460**:98–122.

12. Butlerov AM: **Einiges über die chemische Structur der Körper**. *Zeitschrift für Chemie* 1861, **4**:549–560.

13. Rozenberg G, Ehrig H: *Handbook of graph grammars and computing by graph transformation*, *Volume 1*. Singapore: World Scientific 1997.

14. Andersen J, Flamm C, Merkle D, Stadler P: **Inferring Chemical Reaction Patterns Using Graph Grammar Rule Composition** 2013[http://arxiv.org/abs/1208.3153]. [Accepted, J Sys Chem].

15. Zeigarnik AV: **On Hypercycles and Hypercircuits in Hypergraphs**. In *Discrete Mathematical Chemistry*, *Volume 51 of* DIMACS series in discrete mathematics and theoretical computer science. Edited by Hansen P, Fowler PW, Zheng M, Providence, RI: American Mathematical Society 2000:377–383.

16. Bishop K, Klajn R, Grzybowski B: **The Core and Most Useful Molecules in Organic Chemistry**. *Angew. Chem. Int. Ed.* 2006, **45**:5348–5354.

17. Fialkowski M, Bishop K, Chubukov V, Campbell C, Grzybowski B: **Architecture and Evolution of Organic Chemistry**. *Angew. Chem. Int. Ed.* 2005, **44**:7263–7269.

18. Grzybowski B, Bishop K, Kowalczyk B, Wilmer C: **The 'wired' universe of organic chemistry**. *Nature Chemistry* 2009, **1**:31–36.

19. Karp P, Caspi R: **A survey of metabolic databases emphasizing the MetaCyc family**. *Arch. Toxicol.* 2011, **85**:1015–1033.

20. Weininger D: **SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules**. *J. Chem. Inf. Comput. Sci.* 1988, **28**:31 – 36.

21. Weininger D, Weininger A, Weininger JL: **SMILES 2. Algorithm for Generation of Unique SMILES Notation**. *J. Chem. Inf. Comput. Sci.* 1989, **29**(2):97 – 101.

22. Cordella L, Foggia P, Sansone C, Vento M: **A (Sub) Graph Isomorphism Algorithm for Matching Large Graphs**. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2004, **26**(10):1367.

23. Benkö G, Flamm C, Stadler PF: **A graph-based toy model of chemistry**. *J. Chem. Inf. Comput. Sci.* 2003, **43**(4):1085 – 1093.

24. Ricardo A, Carrigan MA, Olcott AN, Benner SA: **Borate minerals stabilize ribose**. *Science* 2004, **303**:196.

25. Nicolaou K, Snyder S, Montagnon G T amd Vassilikogiannakis: **The Diels-Alder Reaction in Total Synthesis**. *Angew. Chem. Int. Ed.* 2002, **41**:1668–1698.

26. Decker P, Schweer H, Pohlamnn R: **Bioids : X. Identification of formose sugars, presumable prebiotic metabolites, using capillary gas chromatography/gas chromatography-mas spectrometry of n-butoxime trifluoroacetates on OV-225**. *J Chromatogr A* 1982, **244**:281–291.

27. Benner SA, Kim HJ, Carrigan MA: **Asphalt, water and the prebiotic synthesis of ribose, ribonucleosides, and RNA**. *Acc. Chem. Res.* 2012, **45**(12):2025–2034.

28. Benner SA, Kim HJ, Kim MJ, Ricardo A: **Planetary organic chemistry and the origins of biomolecules**. *Cold Spring Harb Perspect Biol* 2010, **2**:a003467.

29. O'Boyle N, Banck M, James C, Morley C, Vandermeersch T, Hutchison G: **Open Babel: An open chemical toolbox**. *Journal of Cheminformatics* 2011, **3**:33.

30. Kim HJ, Ricardo A, Illangkoon HI, Kim JK, Carrigan MA, Frye F, Benner SA: **Synthesis of carbohydrates in mineral-guided prebiotic cycles**. *J. Am. Chem. Soc.* 2011, **133**(24):9457–9468.

31. Oró J, P KA: **Synthesis of purines under possible primitive earth conditions. I. Adenine from hydrogen cyanide**. *Arch Biochem Biophys* 1961, **94**:217–227.

32. Ferris JP, Wos JD, Nooner DW, Oró J: **Chemical evolution. XXI. The amino acids released on hydrolysis of HCN oligomers**. *J. Mol. Evol.* 1974, **3**:225–231.

33. Ferris JP, Joshi PC, Edelson EH, Lawless JGJ: **HCN: a plausible source of purines, pyrimidines and amino acids on the primitive earth**. *J. Mol. Evol.* 1978, **11**:293–311.

34. Voet A, Schwartz A: **Prebiotic adenine synthesis from HCN-Evidence for a newly discovered major pathway**. *Bioorg. Chem.* 1983, **12**:8–17.

35. Miyakawa S, J CH, Miller SL: **The cold origin of life: B. Implications based on pyrimidines and purines produced from frozen ammonium cyanide solutions**. *Origins Life Evol. Biosphere* 2002, **32**:209–218.

36. Saladino R, Crestini C, Costanzo G, DiMauro E: **Advance in the prebiotic synthesis of nucleic acids bases: Implications for the origin of life**. *Curr. Org. Chem.* 2004, **8**:1425–1443.

37. Borquez E, Cleaves HJ, Lazcano A, Miller SL: **An investigation of prebiotic purine synthesis from the hydrolysis of HCN polymers**. *Orig Life Evol Biosph.* 2005, **35**:79–90.

38. Matthews CN, Minard RD: **Hydrogen cyanide polymers, comets and the origin of life**. *Faraday Discuss.* 2006, **133**:393–401 & 427–452.

39. Ritson D, Sutherland JD: **Prebiotic synthesis of simple sugars by photoredox systems chemistry**. *Nat Chem* 2012, **4**:895–899.

40. Andersen J, Andersen T, Flamm C, Hanczyc M, Merkle D, Stadler P: **Navigating the Chemical Space of HCN Polymerization and Hydrolysis: Guiding Graph Grammars by Mass Spectrometry Data** 2013[http://www.imada.sdu.dk/~daniel/hcn2013paper.pdf]. [Under minor revision].

41. increpare games: **Catalan** accessed 04 Feb 2013, [http://www.increpare.com/2011/01/catalan/].

42. Kauffman KJ, Prakash P, Edwards JS: **Advances in flux balance analysis**. *Curr. Opin. Biotechnol.* 2003, **14**(5):491 – 496.

43. Klamt S, Stelling J: **Two approaches for metabolic pathway analysis?** *Trends Biotechnol.* 2003, **21**(2):64 – 69.

A Transformation Rules for the Formose Chemistry

The main formose chemistry consists of two reversible reactions, keto-enol tautomerism and aldol addition. These reaction patterns are listed below as four transformation rules, r_0 to r_3, one for each direction. Additionally, for modeling borate inhibition we use a borate addtion rule, r_4. The inverse of this rule, r_5, is used for generating the underlying molecule without borate. The rules r_6 and r_7 are used for converting between acidic and non-acidic hydrogens in borate complexes. Note that the context graph, K, of r_6 and r_7 also uses the labeling scheme "L label | R label", with the meaning that the vertex changes label from "L label" to "R label".

A.1 r_0, Keto-enol Tautomerism, Keto-to-enol
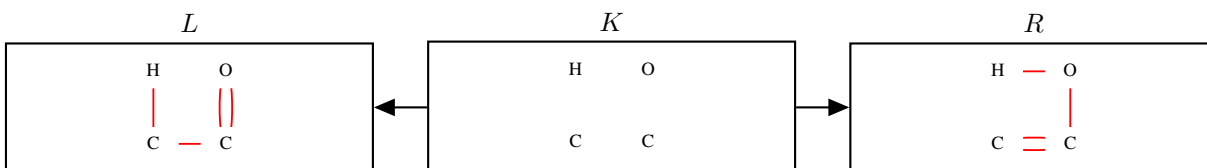
A.2 r_1, Keto-enol Tautomerism, Enol-to-keto
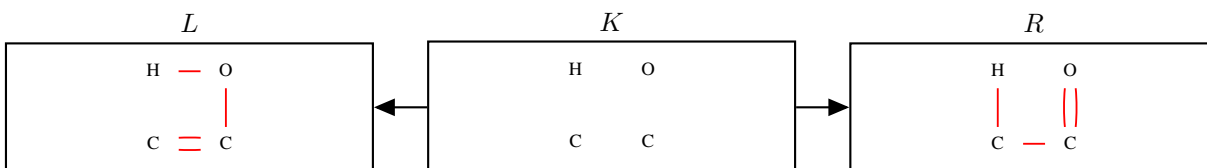
A.3 r_2, Aldol Reaction, Addition
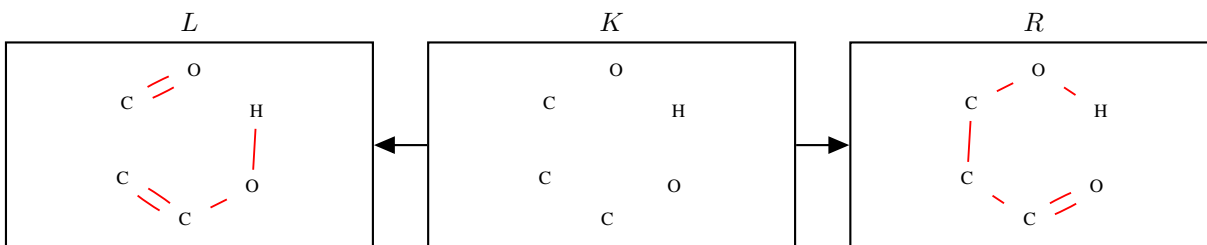
A.4 r_3, Aldol Reaction, Splitting

23
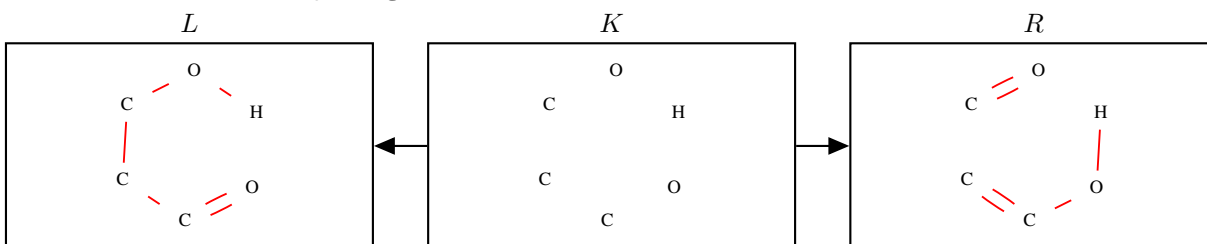
A.5 r_4, Borate Reaction, Addition
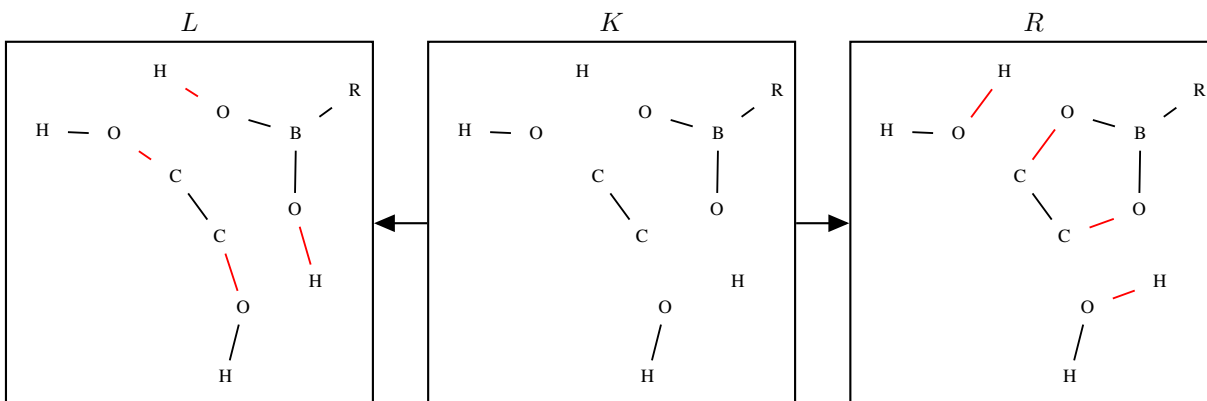
The rule has the following matching condition: none of the adjacent edges of the carbon vertices may represent a double bond.
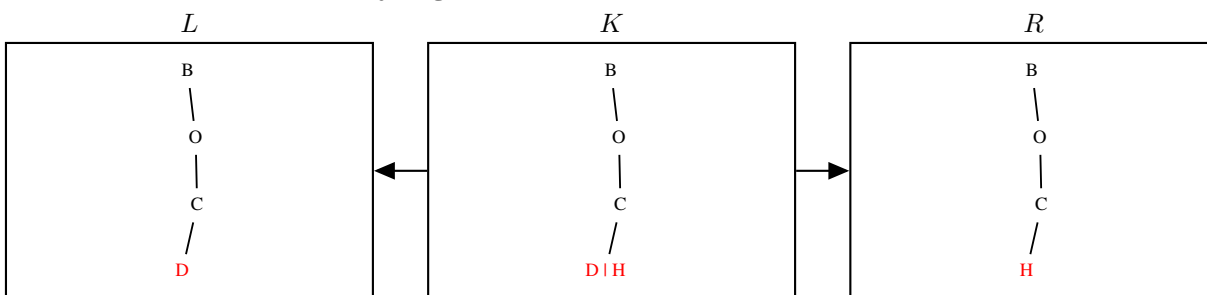
A.6 r_5, Borate Reaction, Splitting
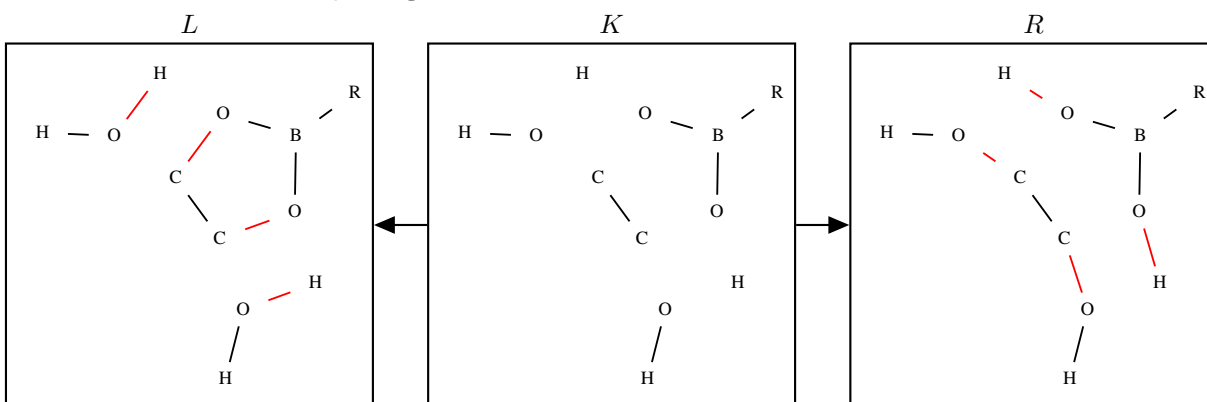
A.7 r_6, Acidic to Non-acidic Hydrogen
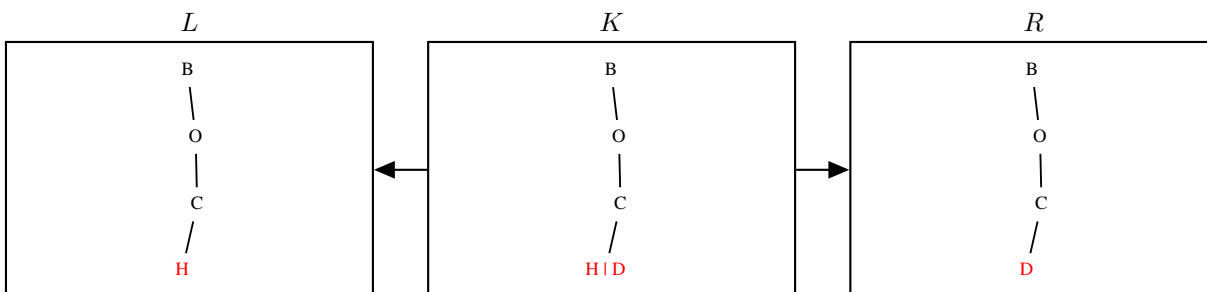
A.8 r_7, Non-acidic to Acidic Hydrogen

L	K	R
B \\ O | C / D	B \\ O | C / D|H	B \\ O | C / H

$L \longleftarrow K \longrightarrow R$

B Additional Diels-Alder Chemistry Figure

Fig. 14 shows the derivation graph obtained from the breadth-first expansion of the Diels-Alder chemistry. The number of expansion steps is only 2.
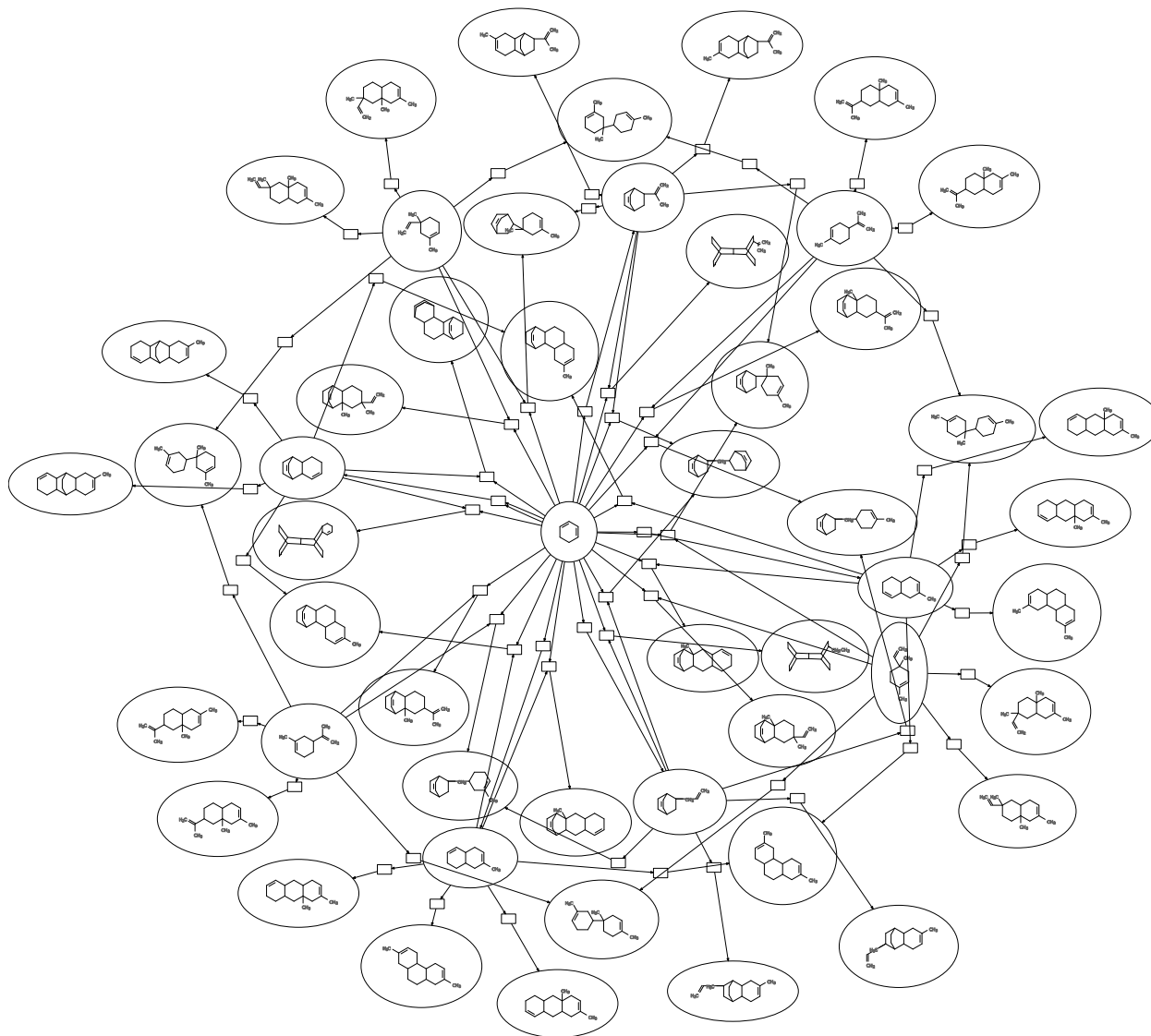
Figure 14: The derivation graph resulting from evaluating the breadth-first expansion strategy Q_{BFS} = addSubset[{isoprene, cyclohexadine}] → repeat[Q_p, 2] (on an empty graph state). To minimize clutter, the vertex with isoprene and the corresponding edges are not shown.

C Solving the Catalan Game

The Catalan game [41] is a puzzle game in which the player in each level is presented with a simple undirected graph without labels. The goal is to transform the graph into a single vertex using the following rewriting rule; given a vertex v with degree exactly 3, identify v with its neighbours and preserve simpleness of the graph by identifying parallel edges and deleting loops. Fig. 15 shows level 1 with the intermediary graphs towards the goal graph with a single vertex.

The transformation in the game can not be formulated as a single rule in the DPO formalism, because such rules must explicitly match the vertices and edges which are changed, while the Catalan transformation needs to change arbitrarily many edges. In the following we show how the strategies can be used to implement a move in the game, using only DPO rules.

Let g be the graph from some Catalan level, with all edge labels set to the empty string and all vertex labels set to the arbitrarily chosen label "0". A high-level description of a move is:

1. Find a vertex v with at least 3 neighbours and mark it by changing the label to "A". Mark the 3 matched neighbours with the label "R".

2. If possible, find another fourth neighbour of v and mark v with "FAIL".

3. Discard all graphs with a vertex with the label "FAIL".

4. For all edges e with both end-vertices having label "R", remove e.

5. For all edges ur with u having label "0" and r having label "R", add uv if it does not exist already and then remove ur.

6. For all edges ur with u having label "0" and r having label "R", remove ur.

7. Remove all neighbours of v having label "R".

8. Unmark v by changing the label to "0".

Step 3 can be implemented with a filtering strategy while the other steps each require a transformation rule. The following strategy can be used to solve a level, in the sense that if a graph with a single vertex with label "0" is found, then a path to that graph is equivalent to a solution. The details of the transformation rules (mark, markForFail, removeInterR, reattachExternal, removeAttached, removeR and unmark) are shown in Appendix D.

$$
\begin{aligned}
Q_{\text{catalan}} = \text{addSubset}[\{g\}] &\to \text{altRuleApp}[\text{repeat}[\\
&\quad \text{mark} \to \text{revive}[\text{markForFail}] \to \text{filterUniverse}[P_{\text{fail}}]\\
&\quad \to \text{repeat}[\text{revive}[\text{removeInterR}]]\\
&\quad \to \text{repeat}[\text{revive}[\text{reattachExternal}]]\\
&\quad \to \text{repeat}[\text{revive}[\text{removeAttached}]]\\
&\quad \to \text{removeR} \to \text{unmark}\\
&]]\\
P(g', F) &\equiv \text{no vertex of } g' \text{ has the label "FAIL"}
\end{aligned}
$$

With strategy Q_{catalan} all 56 levels of Catalan could be solved, all but one level took less than 10 minutes of computation time. Fig. 16b exemplarily shows the derivation graph created when executing the strategy with g encoding level 25 of the game, and Fig. 16a show the initial level graph. The resulting derivation graph is, in contrast to chemical reaction networks, not a hypergraph. However, the graph clearly illustrates subspaces that are connected via a small number of bridging edges. Such subspaces are also expected in chemical reaction networks.
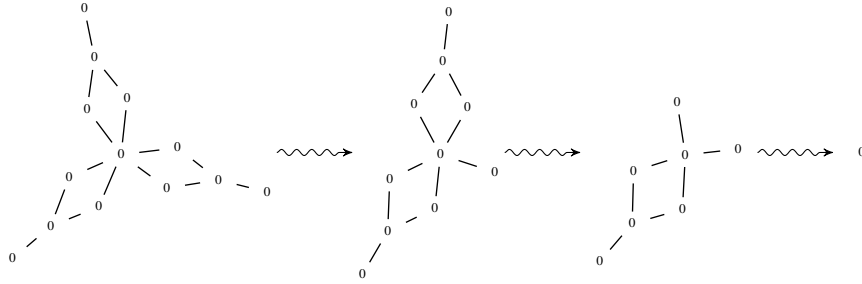
Figure 15: Level 1 of the Catalan game and the intermediary graphs during transformation to a graph with a single vertex.
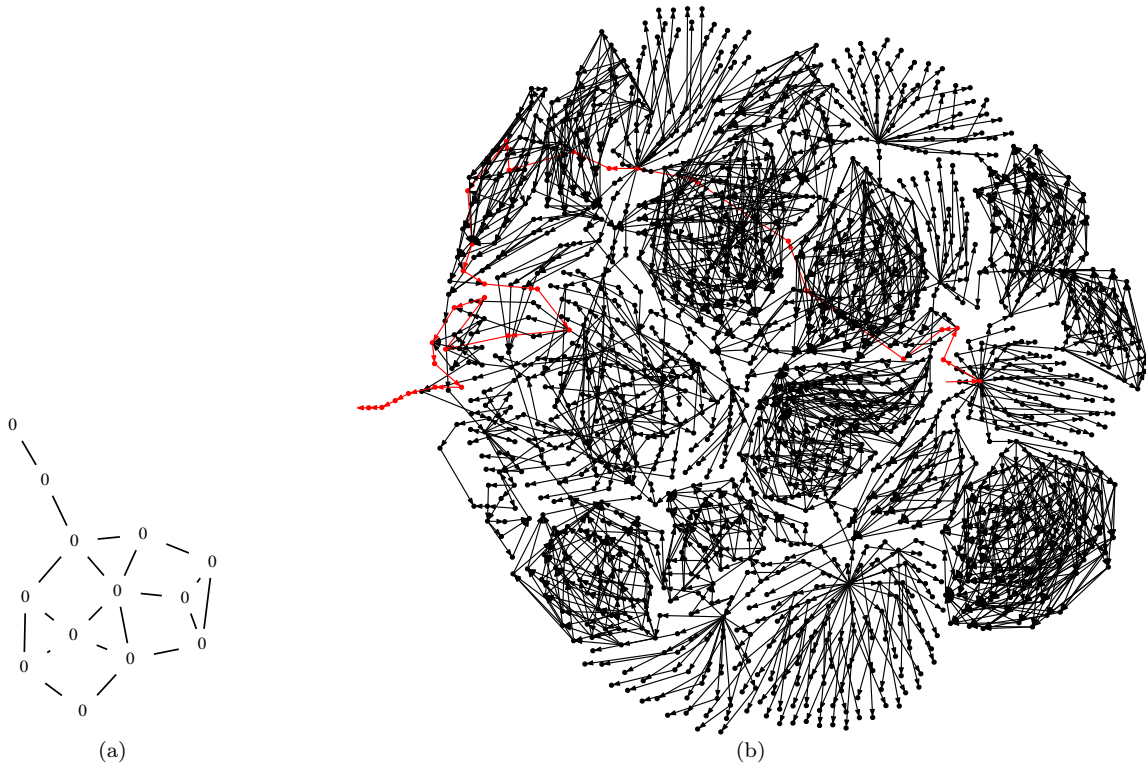

(a)

(b)

Figure 16: The derivation graph created during expansion of level 25 of the Catalan game. A path equivalent to a solution is highlighted.

D   Transformation Rules for the Catalan Game

The following sections contain visualization of the rules used in the strategy to solve a level in the Catalan game. Vertices and edges shown in red are those being changed during transformation. For some vertices the change is only a change of label. The label in the context graph, K, is for those in the format "L | R" with L and R being the label in the left and right side of the rule.
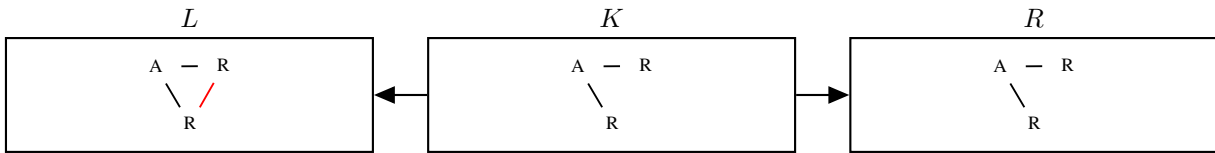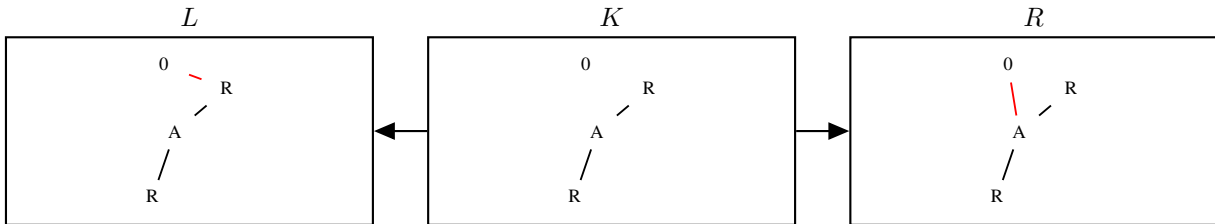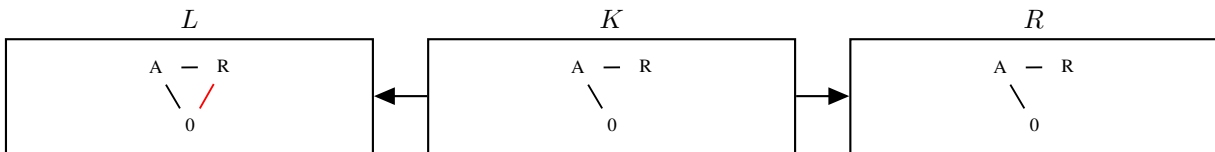
D.1 mark
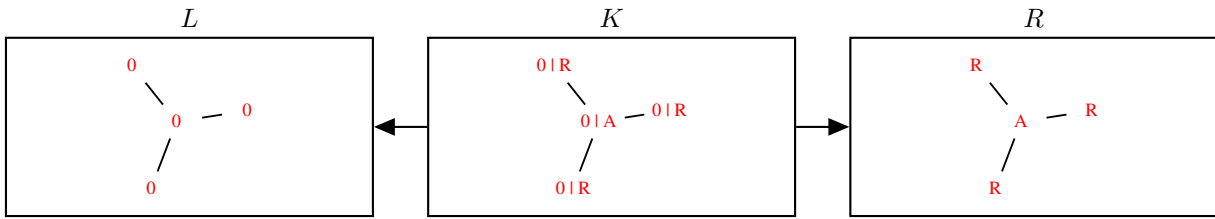
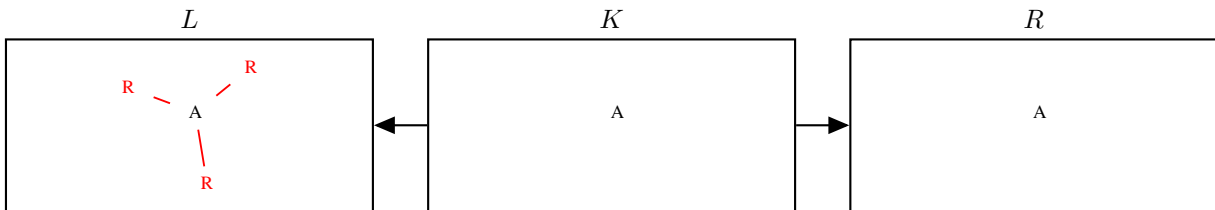

D.2 markForFail
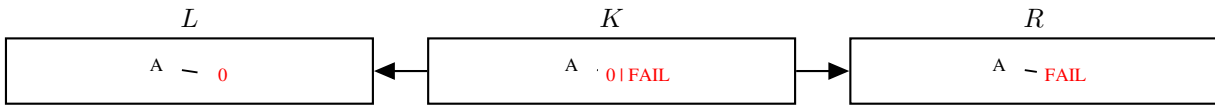
D.3 removeInterR

D.4 reattachExternal

D.5 removeAttached

D.6 removeR

D.7 unmark

L	K	R
A	A|0	0