SFILES 2.0: AN EXTENDED TEXT-BASED FLOWSHEET REPRESENTATION

A PREPRINT

Gabriel Vogel
Department of Chemical Engineering
Delft University of Technology
Delft, Netherlands

Lukas Schulze Balhorn
Department of Chemical Engineering
Delft University of Technology
Delft, Netherlands

Edwin Hirtreiter
Department of Chemical Engineering
Delft University of Technology
Delft, Netherlands

Artur M. Schweidtmann
Department of Chemical Engineering
Delft University of Technology
Delft, Netherlands
a.schweidtmann@tudelft.nl

August 2, 2022

ABSTRACT

SFILES is a text-based notation for chemical process flowsheets. It was originally proposed by d'Anterroches [1] who was inspired by the text-based SMILES notation for molecules. The text-based format has several advantages compared to flowsheet images regarding the storage format, computational accessibility, and eventually for data analysis and processing. However, the original SFILES version cannot describe essential flowsheet configurations unambiguously, such as the distinction between top and bottom products. Neither is it capable of describing the control structure required for the safe and reliable operation of chemical processes. Also, there is no publicly available software for decoding or encoding chemical process topologies to SFILES. We propose the SFILES 2.0 with a complete description of the extended notation and naming conventions. Additionally, we provide open-source software for the automated conversion between flowsheet graphs and SFILES 2.0 strings. This way, we hope to encourage researchers and engineers to publish their flowsheet topologies as SFILES 2.0 strings. The ultimate goal is to set the standards for creating a FAIR database of chemical process flowsheets, which would be of great value for future data analysis and processing.

Keywords Flowsheet graph · Process flow diagram · Artificial intelligence · FAIR data · STRING notation

1 Introduction

Chemical process flowsheets, also known as process flow diagrams (PFDs) [2, 3], are the current standard for depicting and communicating the topology of unit operations in chemical processes (see Figure 1a). PFDs are used in industry and academia during conceptual process design and consequently there exists at least one PFD for every chemical process in the world. Besides process flow diagrams, Piping and Instrumentation Diagrams (P&IDs) [2, 3] are a central representation class of chemical processes. They include additional information about instrumentation, valves, control structures, and piping [4]. Due to contained process-specific knowledge P&IDs provide valuable details for a deep understanding of the chemical process. Therefore, P&IDs are interdisciplinary employed at every stage of a chemical plant: from engineering and design, to hazard and operability studies (HAZOP), to operation and tracking changes during maintenance [5]. Currently, PFDs and P&IDs are usually drawn in computer programs and exported as images or PDF documents. Despite some recent efforts in Smart P&IDs and open data exchange formats [6], it seems that the information content of flowsheet diagrams in documents often remains inseparable from the medium, like hieroglyphs carved in stone. The main reason for this development is that PFDs and P&IDs in the form of images or PDFs are widely utilized as an interdisciplinary communication tool for easily exchanging first process ideas, but also advanced plant designs between experts from different domains (e.g. process engineers, material scientists, management, etc.). Also, proprietary process simulation software often does not facilitate interoperability and data exchange. However, the document-based communication of flowsheet information hinders the development of findable, accessible, interoperable, and reusable (FAIR) [7] data. This also has consequences for the use of advanced data analysis and data processing tools. Currently, some aspects of chemical process design can be tedious and repetitive, while FAIR process data could enable automated data processing. In our previous work, we also argue that the lack of structured data is a major hurdle for advances of artificial intelligence in chemical process engineering [8].

Chemical flowsheets can be represented as directed graphs [9, 10]. The flowsheet graph (see Figure 1b) consists of nodes that represent the unit operations and directed edges that represent the stream connections. Graphs are computationally accessible and further offer the possibility to store additional process information as node or edge attributes. However, using the graph as flowsheet representation usually requires knowledge of programming languages and graph libraries, both for the process designer and for engineers who want to reuse the flowsheet.

Text-based representations are a promising alternative to graph representations for the communication of flowsheet information. In 2006, d'Anterroches [1] proposed the Simplified Flowsheet Input-Line Entry-System (SFILES) which is a text-based notation to represent flowsheet topologies. The SFILES is inspired by the Simplified Molecule Input-Line Entry-System (SMILES) [11] notation, which has become a standard storage and exchange format for molecules. Using SFILES as flowsheet storage and exchange format brings several advantages compared to images and graphs. Standardization of the text-based representation is one advantage over flowsheet images that usually vary due to different drawing software. Furthermore, the text-based representation is an efficient exchange format that can be included in publications and directly used for data analysis and processing, which sets it apart from the graph representation.

SFILES have already enabled the development of advanced data processing techniques on flowsheets. Tula et al. [12, 13] used it to compare process flowsheets for a given synthesis problem. Their approach enabled them to find more sustainable process alternatives. In other work, the SFILES notation was slightly modified and used for pattern recognition in chemical process flowsheets [9, 10]. With the help of sequence alignment algorithms, the authors successfully identified common design patterns in chemical process flowsheets. Nevertheless, previous work does not include a complete description of the connectivity and the stream paths when dealing with unit operations with multiple in- and outlet streams, i.e., the distinction between top and bottom products or stream paths through multi-stream heat exchangers. Furthermore, the SFILES notation in previous work is limited to PFDs, neglecting important information contained in P&IDs, such as control structures. To the best of our knowledge, there is also no publicly available software for the automated conversion between flowsheet graphs and SFILES 2.0 strings.

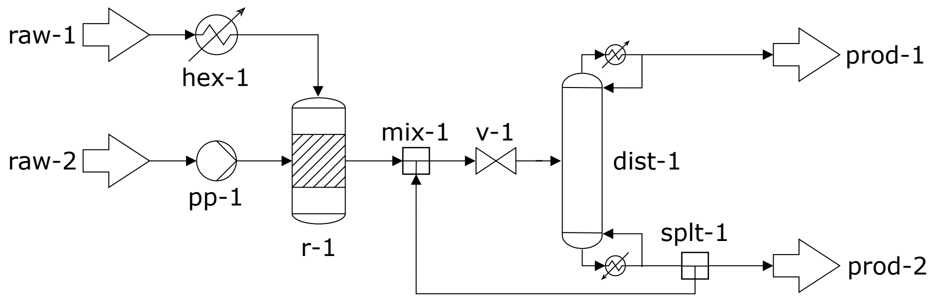
In this work, we propose the SFILES 2.0 and provide a comprehensive description of the extensions and modifications compared to previous work. Moreover, we suggest naming conventions to pave the way toward standardized SFILES strings. The extensions in this paper include a set of rules for the flowsheet graph representation, specifying a new way to unambiguously represent multi-stream heat exchangers and unit operations with top and bottom in- and outlet streams in the flowsheet graph. Subsequently, we modified and extended the original SFILES notation rules, which allow an unambiguous string representation and enable a reversible conversion between a flowsheet graph and its corresponding SFILES 2.0 string. Eventually, it should be possible to describe flowsheet topologies of higher complexities while still encoding all necessary topological information in the SFILES 2.0 string. Additionally, we address the inclusion of control structures contained in P&IDs in the flowsheet graph and SFILES 2.0 notation. Moreover, we implemented a conversion algorithm in Python and made it openly accessible in a GitHub repository [14] with illustrative examples, encouraging researchers to publish their future chemical process flowsheets with the corresponding SFILES 2.0 strings. This way, we hope to contribute to creating and continuously extending a machine-readable SFILES 2.0-based database of chemical process flowsheet topologies.

2 Background

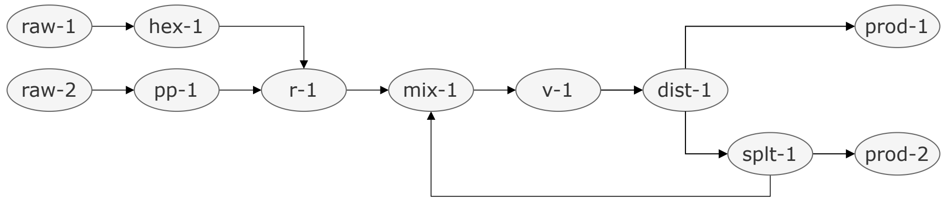
The following outlines previous work on the flowsheet graph representation and SFILES notation rules, which lays the foundation for our work.

2.1 Flowsheet graph representation

A graph is a data structure that consists of nodes, also called vertices, and edges. Edges are connections between nodes and can be either directed or undirected, defining whether the graph is directed or undirected. The original description of the SFILES string [1] uses a directed flowsheet graph with process groups as nodes and the connections between these process groups as edges. The process groups can either represent one unit operation or a set of unit operations. Herein, we focus on single unit operations in flowsheets, similar to the work of Zhang et al. [9] defining unit operations as nodes and the connecting streams as edges. Figure 1a shows an exemplary flowsheet with two inlet streams, a reactor, a distillation column (reboiler and condenser included), a recycle of the bottom product, and two product streams. The used abbreviations are based on the standardized unit operation names in Table 2. When constructing the corresponding flowsheet graph in Figure 1b, the nodes need to be numbered to obtain a unique definition of nodes and their associated edges. We can distinguish the graph nodes using their in- and out-degree, whereby the in-degree is the number of edges directed towards a node, and the out-degree is the number of edges directed away from a node. Inlet nodes with the name `raw` always exhibit an in-degree=0, and outlet nodes with the name `prod` always have an out-degree=0. A node with an in-degree>1 means that graph branches are converging at that node (in Figure 1b `r-1, mix-1`), and a node with an out-degree>1 indicates a new branching at the considered node (in Figure 1b `dist-1, splt-1`).



(a)



(b)

Figure 1: (a) Simple chemical process flowsheet with branches and one recycle stream. (b) Graph representation of the flowsheet in (a).

2.2 Original SFILES notation

The original SFILES notation rules [1] are outlined in this section using the flowsheet graph in Figure 1b. Starting with the inlet node raw-1 the corresponding SFILES string of this flowsheet graph is

```
(raw-1)(hex-1)(r-1)[<(pp-1)<(raw-2)](mix-1)<1(v-1)(dist-1)[(prod-1)](splt-1)1(prod-2).
```

Process groups, or in this example, abbreviations for the unit operations are noted in parenthesis. The SFILES string is read from left to right and two consecutive unit operations in parenthesis imply a connection, e.g., `(raw-1)(hex-1)` implies a connection from `(raw-1)` to `(hex-1)`. In the case of branching in the graph, e.g., after the distillation system (dist-1) in Figure 1a, all except the last considered branch during the conversion from flowsheet graph to string (see Section 4.1), are noted in square brackets. In the case of converging branches at a node with an in-degree>1, d'Anterroches [1] uses square brackets and < for backward connections in the SFILES string. Converging branches always occur when the described chemical process comprises multiple input streams. Consequently, the sequence `(r-1)[<(pp-1)<(raw-2)]` implies the connections from `(raw-2)` to `(pp-1)` and `(pp-1)` to `(r-1)`. The last important notation rule applies to recycle connections, such as the one from `(splt-1)` back to `(mix-1)`. Similar to cycles in molecules in the SMILES notation, a number # is used to indicate the start of a recycle (here: (splt-1)1), and <# is used to indicate the end of the directed recycle connection (here: (mix-1)<1). Given the flowsheet graph, the SFILES string generation consists of two steps [1]:

1. Calculation of a unique graph invariant.
2. SFILES generation by traversing the graph with initial node selection and branching decisions based on the graph invariant.

The graph invariant calculation is based on the flowsheet graph structure and is used to assign a unique rank to each node (see Section 4.1). Based on the node ranks, an initial node for the graph traversal is chosen and branching decisions are made. This ensures the generation of a unique SFILES string.

The numbers in a SFILES string are adopted from the node names in the flowsheet graph but do not contain any essential process knowledge. For this reason, in previous work [9, 10] for pattern recognition in flowsheets, the authors used a generalized version of the SFILES string without the unit operation numbers. Removing the numbering in the SFILES string of the example in Figure 1b yields the generalized SFILES

```
(raw)(hex)(r)[<(pp)<(raw)](mix)<1(v)(dist)[(prod)](splt)1(prod).
```

3 SFILES 2.0

In this section, we describe our proposed modifications and extensions of the original SFILES notation. We call this modified version SFILES 2.0. Section 3.1 clarifies minor modifications of the syntax and proposes extensions to unambiguously represent multi-stream heat exchangers and unit operations with complex connectivity, such as separation columns. Thereafter, Section 3.2 describes the notation details that are required to represent the control structure contained in P&IDs. Finally, we propose standardized naming conventions for commonly used unit operations in Section 3.3. In the following, we use generalized SFILES as the standard notation (see Section 2.2).

3.1 Extension of notation

For complex chemical processes, the corresponding flowsheets can get quite large, containing a high number of unit operations and process branches. For a more robust notation of complex converging branches (multiple input streams), we suggest the following modification: When reaching a node with an in-degree>1 during the graph traversal (see Section 4.2), the original SFILES definition uses a backward notation containing < signs for converging branches. In the SFILES 2.0, we note converging branches surrounded by <&| and |, whereby we insert an additional &-sign next to the node that is connected to the considered node with an in-degree>1. Using this notation for the example in Figure 1b yields the generalized SFILES

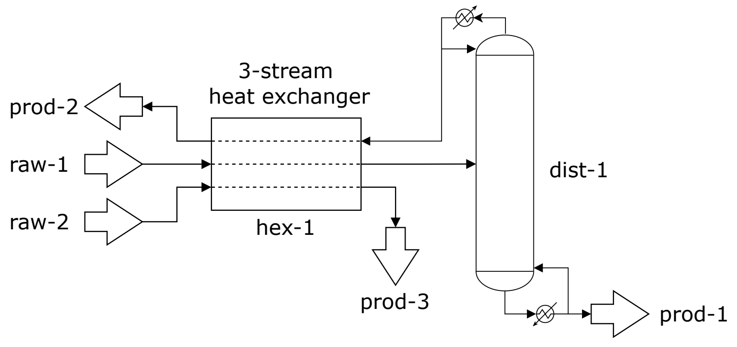
```
(raw)(hex)(r)<&|(raw)(pp)&|(mix)<1(v)(dist)[(prod)](splt)1(prod).
```

It eliminates the backward notation containing < signs and, more importantly, enables a more robust notation of complex converging branches that consist of several branches themselves. An example that illustrates the necessity of this modification is shown in Appendix A.

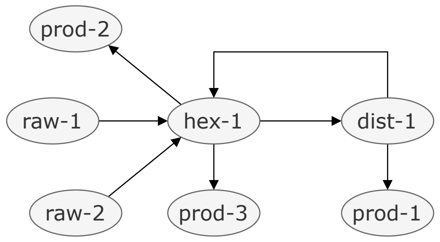
The following extensions in the SFILES 2.0 compared to previous work focus on how to describe the connectivity and the stream paths when dealing with unit operations with multiple in- and outlet streams. A common process characteristic that illustrates the importance of the connectivity information is heat integration, resulting in multi-stream heat exchangers. For instance, cryogenic processes such as air separation often comprise multi-stream heat exchangers. Other examples exhibiting complex connectivity are distillation columns with top and bottom products or even several inlet and outlet streams. The information on how the different streams are connected to the unit operations and are further processed is essential and must be included in the SFILES 2.0 string to enable a reversible reconstruction of the flowsheet graph. The example process in Figure 2 consists of a 3-stream heat exchanger and a distillation column with top and bottom products. Essential information, in this case, is that the inlet `raw-1` is connected to the column via the heat exchanger and the top product is returned to the heat exchanger. Converting the flowsheet to a directed graph and to the generalized SFILES string without connectivity information yields

```
(raw)(hex)<1<&|(raw)&|[(prod)][(prod)](dist)1(prod).
```

Using this SFILES string for the conversion back to a flowsheet graph would be ambiguous in terms of tracking the stream paths through the heat exchanger and the information of which separation product is heat-integrated with the heat exchanger.



(a)



(b)

Figure 2: Flowsheet with complex connectivity characteristics. (a) PFD, (b) graph representation

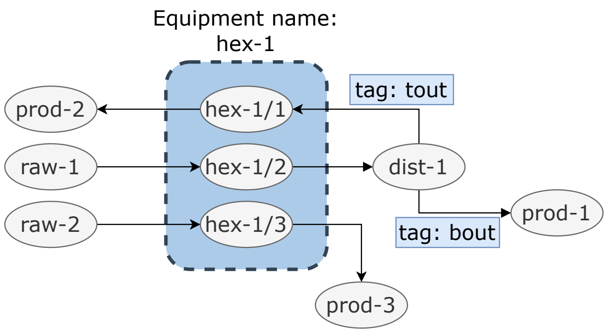


Figure 3: Flowsheet graph with modified node structure of heat exchanger and connectivity attributes for distillation column

There are several possibilities to include the necessary information in the SFILES. Our strategy starts with modifying the flowsheet graph representation derived from the concept of State-Equipment Networks (SEN) [9, 15] for the representation of the superstructure of chemical processes. As shown in Figure 3, we replaced the heat exchanger node with three single nodes that represent the accommodated streams in the heat exchanger equipment. Each node represents one heating or cooling task in that heat exchanger, meaning that the streams are not in direct contact but only transfer heat. We distinguish the node names in the graph by adding a `/#`. Consequently, it is possible to have multiple separate mass trains resulting in multiple unconnected sub-graphs in the flowsheet graph. For instance, one sub-graph for the main process and one sub-graph for a refrigeration cycle. We will use the prefix `n|` in the SFILES string to indicate an independent mass train. In our example, one independent mass train is the connection from `raw-2` through `hex-1/3` to `prod-3`. In the numbered SFILES string, the node names of the heat exchangers contain the heat integration information. In the generalized SFILES string, we need to add this information after removing the numbers. Zhang et al. [9] used the recycle notation for heat integrated heat exchangers. However, the streams in heat exchangers do not mix, hence, formally this is not a recycle and we propose an alternative notation. Next to each heat exchanger node of the same heat exchanger equipment, we insert the same number # in braces (`{#}`). In the case of heat exchangers (heaters and coolers) without heat integration (node has in-degree=1 and out-degree=1), we do not encode this information. Including the new rules for multi-stream heat exchangers, the following string results:

`(raw)(hex){1}(dist)[(prod)](hex){1}(prod)n|(raw)(hex){1}(prod).`

We also need to encode additional information for other unit operations, such as distillation columns with at least one top and one bottom product as outlet streams. We use tags in the SFILES string to indicate the top product branch and the bottom product branch. The difference between, for example, a column and a splitter is that the branched streams after a splitter have the same properties, whereas this, in general, does not hold for separation units. As a result, it is crucial information of the flowsheet topology which process branch results from which separation product. We will use braces to encode that additional connectivity information in the following manner. Given a graph edge from node u to node v with a stream tag x, the connection will be noted as 1. in case of a normal connection, 2. in case of branching, 3. in case of a recycle, and 4. in case of a converging branch.

1. `(u){x}(v)`
2. `(u)[{x}(v)]` or `(u)[...]{x}(v)`
3. `(v)<1...(u){x}1`
4. `(v)<&|...(u){x}&|`

1. 그냥 연결
2. 대괄호를 통해 나가는 분기 표현
3. <#, #를 통해 머터리얼 재순환 표현
4. <&|, &, |를 통해 들어오는 분기 표현
* : n|를 통해 아예 다른 flow 표현
** : <_#, _#f를 통해 제어 흐름 표현 for P&ID

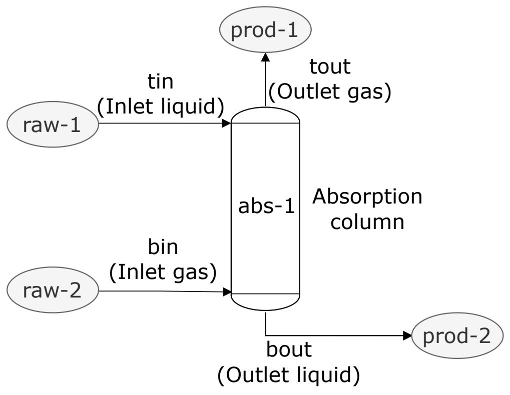
The stream tags must be saved as edge attributes in the flowsheet graph, e.g., the top and bottom outlet stream tags in Figure 3. Combining the rules related to multi-stream heat exchangers and stream tags, the final SFILES 2.0 string results in

```
(raw)(hex){1}(dist)[{bout}(prod)]{tout}(hex){1}(prod)n|(raw)(hex){1}(prod).
```

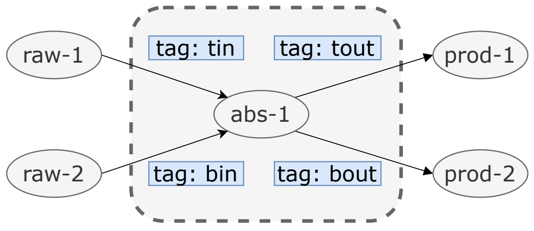
The SFILES 2.0 string now enables the reconstruction of the flowsheet graph without loss of information and ultimately the reproduction of the PFD in Figure 2.

The stream tags can also be applied to other unit operations such as absorption or extraction columns. Figure 4a shows an absorption column with two inlet and two outlet connections. The necessary topological information is contained in the tags `{bin}`, `{tin}`, `{tout}`, and `{bout}`, which are stored as edge attributes in the flowsheet graph in Figure 4b. The SFILES 2.0 string for this hypothetical process is

```
(raw){bin}(abs)<&|(raw){tin}&|[{tout}(prod)]{bout}(prod).
```



(a)



(b)

Figure 4: (a) Absorption column with two inlets and two outlets. (b) Flowsheet graph of (a) with connectivity stream tags.

The same can be applied to all other units operation nodes where the connectivity information is considered essential for the flowsheet topology. Table 1 lists the defined tags.

증류탑같은 2개의 input과 output을 고려하기 위해 bin tin bout tout tag가 고안되었다

Table 1: Set of stream tags in SFILES 2.0

Connectivity information	Stream tag in flowsheet graph and SFILES 2.0
Bottom inlet	bin
Top inlet	tin
Bottom outlet	bout
Top outlet	tout

3.2 Description of control structures

To extend the described text-based notation of PFDs to P&IDs, a representation of the control structure is required. There are three important cases to consider for this: (i) A sensor on a stream controlling a unit operation, (ii) a sensor on a unit operation controlling another unit operation, and (iii) cascading sensors. We introduce the SFILES 2.0 notation for control structure by three illustrative examples in Figure 5. The first example (i) in Figure 5 (a) consists of a sensor measuring the flow rate of a stream and controlling the subsequent valve with this information. Since material streams are implicitly represented in the SFILES 2.0 notation, the measurement of stream information is included by adding the control unit (abbrev. C) between the two unit operations (here `raw` and `prod`), where the state of a stream is required. The control unit is stored as a node like a unit operation. The type of the control unit, which is indicated in the P&ID with a letter code (acc. to DIN EN 62424) [16], is stored in braces next to the node (here {FC} for flow control). Similar to material recycle connections, we represent signal connections to previous unit operations with <_# and _#. The underscore is used to easily distinguish material recycles and signal connections. Furthermore, we use upper case letters for control elements to illustrate the difference to unit operations. These notation rules result in the following generalized SFILES 2.0 for Figure 5 (a):

```
(raw)(C){FC}_1(v)<_1(prod).
```

The second example (ii) in Figure 5 (b) shows a tank whose level is controlled. The direct connection of the instrument to the unit operation is represented as branching at the corresponding node. In the same way as for the first example, the letter code of the control unit is stored as a tag and the instrument is connected to the valve using the signal connection terminology:

```
(raw)(tank)[(C){LC}_1](v)<_1(prod).
```

The third example (iii) in Figure 5 (c) of a control cascade illustrates a combination of the first two cases. The level of the tank is transmitted to a flow controller, which regulates a subsequent valve. The flow transmitter is represented as a branching node at the corresponding unit operation and the flow controller is placed between the tank and valve since its task is to measure the flow rate between the tank and the valve. The connection of the two instruments and the valve is represented with the signal connection notation. Tags store again the letter code of the control units. This results in the following generalized SFILES 2.0 string for Figure 5 (c):

```
(raw)(tank)[(C){LT}_1](C){FC}_2<_1(v)<_2(prod).
```

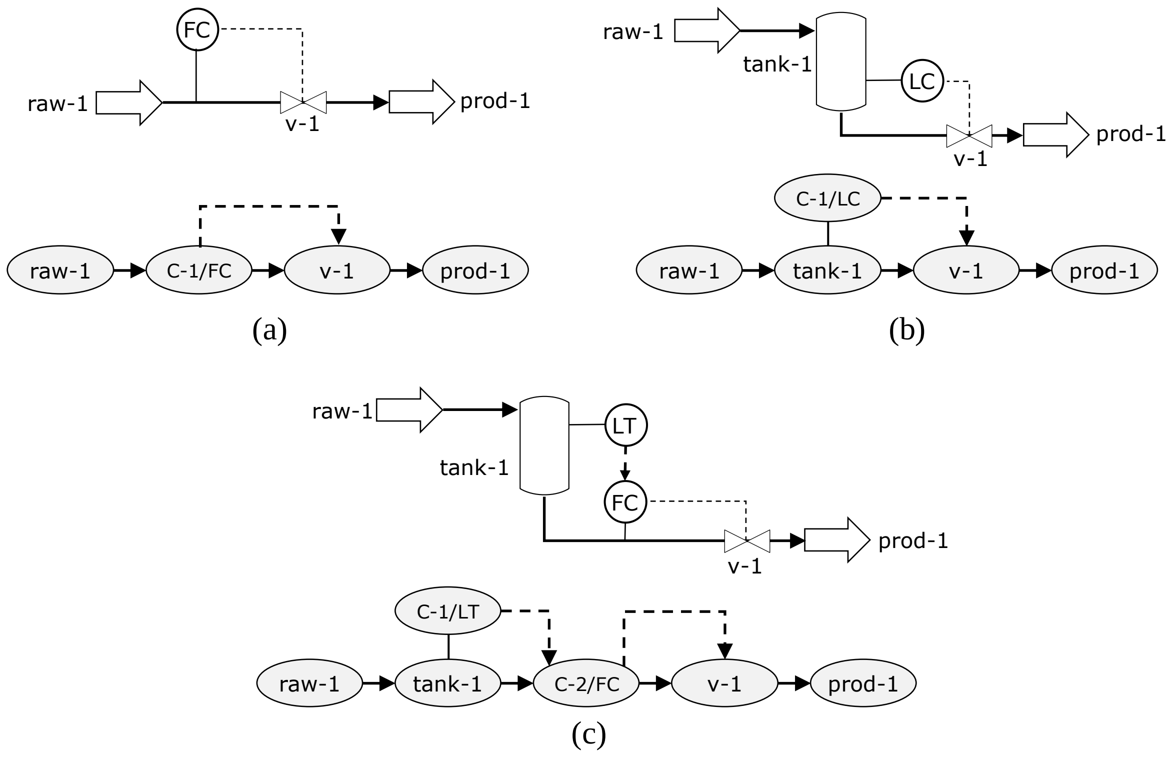


Figure 5: PFD and flowsheet graph of simple control loops. (a) Flow control of material stream, (b) Level control of tank, (c) Level control of tank with control cascade

3.3 Unit operations

This section provides an overview of unit operations in chemical process flowsheets and the abbreviations used in the SFILES 2.0. The selection of unit operations in Table 2 represents commonly used unit operations and is based on the ontology OntoCAPE [17]. Some of the terms in Table 2 are a refined classification of the OntoCAPE ontology, which we performed to include more specific unit operation categories. With increasing access to more flowsheet data, the list of unit operations might need further extension or refinement. The naming conventions, i.e., the abbreviations, should also be followed in the flowsheet graph construction when using the provided code for the conversion from a flowsheet graph to its corresponding SFILES 2.0 string.

SFILES 2.0 string을 보고 각각이 어떤 unit의 줄임말인지 이 표를 통해 알 수 있다

Table 2: Unit operations and abbreviations in SFILES 2.0 based on OntoCAPE ontology [17]

Unit operation	Abbreviation	OntoCAPE term	Typical in-degree	Typical out-degree
Absorption	abs	AbsorptionColumn	2	2
Blower	blwr	Blower*	1	1
Centrifugation	centr	CentrifugationUnit	1	2
Compressor	comp	Compressor*	1	1
Condenser (incl. splitting)	cond	Condenser	1	2
Control unit	C	Control	≥1	≥0
Cyclone	cycl	Cyclone	1	2
Distillation (incl. reboiler and condenser)	dist	DistillationSystem	≥1	≥2
Electrical gas cleaning	egclean	ElectricalGasCleaningUnit	1	2
Expander	expand	Expander*	1	1
Extraction	extr	ExtractionUnit	2	2
Flash	flash	FlashUnit	1	≥2
Gas filtration	gfil	GasFilter	1	2
Hydrocyclone	hcycl	Hydrocyclone	1	2
Heat exchanger	hex	HeatExchanger	≥1	≥1
Liquid filtration	lfil	LiquidFilter	1	2
Mixing	mix	MixingUnit	≥1	1
Orifice plate	orif	OrificePlate*	1	1
Pipe	pipe	Pipe*	1	1
Pump	pp	Pump*	1	1
Product stream	prod	OutputProduct	1	0
Reactor	r	ChemicalReactor	≥1	≥1
Raw material	raw	RawMaterial	0	1
Reboiler (incl. splitting)	reb	Reboiler	1	2
Rectification (incl. reboiler and condenser)	rect	RectificationSystem	≥1	≥2
Scrubbing	scrub	Scrubber	2	2
Separation (no further sub-specification)	sep	SeparationUnit	≥1	≥2
Splitting	splt	SplittingUnit	1	≥2
Stripping	strip	StrippingSystem*	2	2
Storage	tank	StorageUnit*	≥0	≥1
Valve	v	Valve*	1	1
Unknown unit operation	X	-	-	-

*This term is part of our extension of the OntoCAPE ontology and not in the original OntoCAPE documentation.

3.4 Limitations of SFILES 2.0

Nevertheless, there remain limitations of the SFILES 2.0 notation in the case of very complex process topologies. In the set of standardized stream tags for separation columns, we only consider top and bottom in- and outlets. The latter certainly covers the most common arrangements of unit operations in processes. Still, more complex examples such as the air separation process can contain columns with far more than two in- and outlets, respectively. For such complex unit operations, the current SFILES 2.0 notation rules do not suffice to ensure a reversible conversion between the SFILES string and the flowsheet in terms of the order of in- and outlets. At this point, we would like to mention that all types of flowsheets can be converted to an SFILES string. However, with a possible loss of information due to missing tags and, therefore, no fully reversible conversion back to the actual flowsheet. Theoretically, it would be possible to extend the notation to encode more complex information, e.g., by changing the stream tags to positions relative to the height of columns (between 0 and 1, e.g., `{1.0_out}` for the top outlet). Another approach could be to further divide equipment into several nodes, similarly to the SEN-based method for multi-stream heat exchangers. The braces notation could optionally also store flowsheet information beyond the topology in the SFILES 2.0. For instance, additional stream-related process information like the pressure, temperature, or components can be stored as edge attributes.

Additionally, information describing a unit operation, such as the geometrical dimensions or operating conditions are currently not stored in the SFILES 2.0 string. When desired, it could be stored as node attributes in the flowsheet graph and included in braces within the parentheses notation for unit operations. However, in this context, it must be pointed out that this information results in continuous variables which are not essential for describing the topology of flowsheets. Furthermore, a more detailed description of the control structure, e.g., whether the instrument is a field-mounted or shared display device, is currently not provided.

4 SFILES 2.0 generation algorithm

SFILES 2.0 string <==> flowsheet graph
변환 어떻게 하는지 설명

This section describes the conversion algorithm between flowsheet graphs and SFILES 2.0 strings. Our implementation consists of the conversion algorithm from flowsheet graphs to SFILES 2.0 strings as well as the algorithm for the conversion of SFILES 2.0 strings to the corresponding flowsheet graphs and is publicly available in a GitHub repository [14]. Similar to the original SFILES notation algorithm (see Section 2.2), the two major steps for the SFILES 2.0 string generation are the determination of the graph invariant (Section 4.1) and the graph traversal (Section 4.2). If a control structure is present in the flowsheet graph the nodes of the control units are treated as unit operation nodes. Only the signal connections (dashed line in P&IDs) are removed before determining the graph invariant and the graph traversal, to ensure complete interoperability between SFILES 2.0 generated from P&IDs and PFDs. The signal connections are added afterward using the notation mentioned in Section 3.2.

4.1 Determination of graph invariant

graph --> string :
1. 그래프를 유일한 형태로 변환 후
2. 순회하는 순서대로 string을 생성

The graph invariant aims to yield a unique rank for each node. The determination of this graph invariant is also known as graph canonization. The first step in our implementation is based on the Morgan algorithm [18], similarly to the description in Zhang et al. [9]. As illustrated in Figure 6, the initialization starts with assigning all nodes a corresponding node value of 1. Next, each node value is updated with the sum of all neighbor's node values. After the first update, the node values equal their connectivity in the graph. This step aims to increase the variable `val_set` which is defined as the number of unique node values in the graph. The procedure is repeated until `val_set` does not increase for `max_iter` iterations. Finally, the nodes are ranked based on their values. In case there are multiple sub-graphs, as described in Section 3.1, the graph invariant is determined for both graphs separately. The sub-graph with fewer nodes will be assigned a lower priority and noted last in the SFILES 2.0 string. The Morgan algorithm does not yield unique ranks in all cases. Especially in the case of symmetric graphs, there are often multiple nodes with the same value. However, the SFILES 2.0 string generation algorithm requires all nodes to have a unique rank. For this reason, we introduce a rule-based approach for breaking the ties of equally ranked nodes. We use the following procedure to break the ties.

1. Rank (small is higher priority): Control node < Outlet node < Inlet node < Other nodes
2. Rank according to the number of successors[1] in the graph
   (a) Outlet/Control nodes: does not apply
   (b) Inlet nodes: the higher the number of successors the lower the rank
   (c) Other nodes: the lower the number of successors the lower the rank
3. String comparison (smaller rank for earlier appearance in alphabet) of equally ranked node names (unit operation abbreviations) and associated edges

[1]The length of the depth first search tree of the node in the graph is used.

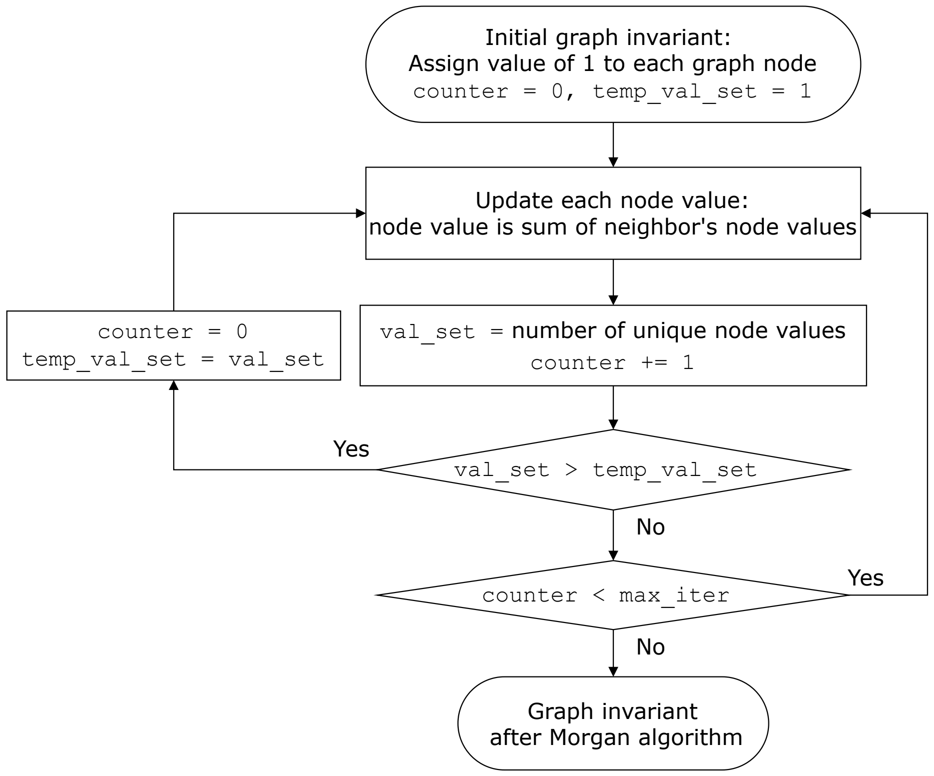


Figure 6: Morgan algorithm for graph invariant determination

4. Ranking by graph node (unit) numbering

In steps 1-3, we only use the generalized SFILES because the SFILES string should only be dependent on the intrinsic graph structure but not the numbering of the unit operations. Step 2 is subdivided into inlet and other nodes to improve the readability of the resulting SFILES string. Nodes still tied after step 3 can be exchanged arbitrarily without a resulting change in the generalized SFILES string. Therefore, in step 4, the nodes are ranked by their unique node names with unit numbering. Table 3 shows the node ranking for the example in Figure 1b.

Table 3: Node ranks for flowsheet graph in Figure 1b

`raw-1`	`raw-2`	`prod-1`	`prod-2`	`hex-1`	`pp-1`	`v-1`	`dist-1`	`r-1`	`splt-1`	`mix-1`
1	2	3	4	5	6	7	8	9	10	11

4.2 Graph traversal

The SFILES string results from traversing the graph after determining its invariant. We will use the depth-first search (DFS) algorithm to traverse the flowsheet graph and write the SFILES string. Starting from an initial inlet node, the DFS algorithm explores the graph branches sequentially as far as possible (until reaching an outlet node or previously visited node) before backtracking to the last branching point. Both the initial node selection as well as the branching decisions are made based on the node ranking, i.e., nodes with lower ranks are selected first. In the case of multiple inlet nodes or sub-graphs, one DFS traversal does not visit all nodes. To mitigate this problem a virtual node is inserted to which all initial nodes (in-degree=0) are connected. Since cycle processes do not exhibit a distinct initial node, the node with the lowest rank, which is not an outlet node (out-degree=0), is selected and connected to the virtual node. After ensuring that every node present in the flowsheet is linked to the virtual node, one graph traversal starting from the virtual node is sufficient.

Using the example in Figure 1b, we will explain how the DFS algorithm and the SFILES string generation work. According to Figure 1b, the nodes `raw-1` and `raw-2` with an in-degree=0 are connected to the virtual node and the graph traversal is started from there. Since `raw-1`, according to Table 3, has the lowest rank, the DFS visits this inlet node first. The successor nodes, in specific `hex-1, r-1, mix-1, v-1, dist-1`, are visited one after another and noted in parentheses. After `dist-1` the top branch continues with `prod-1` (rank 3) and thereafter the bottom branch with (`splt-1`) (rank 10). Thus, the top branch starting with `prod-1` is visited first. The bottom branch leads to the mixer and after the second product `prod-2`, the first graph traversal ends. The resulting generalized SFILES 2.0 string is:

```
(raw)(hex)(r)(mix)<1(v)(dist)[{tout}(prod)]{bout}(splt)1(prod).
```

The next node for the second graph traversal from the virtual node is `raw-2`. The branch converges in the reactor node `r-1` and the final generalized SFILES string is

```
(raw)(hex)(r)<&|(raw)(pp)&|(mix)<1(v)(dist)[{tout}(prod)]{bout}(splt)1(prod).
```

Cycle processes are a special case of flowsheet topologies with no inlet nodes (in_degree=0). The cycle process can be either the complete flowsheet graph or a sub-graph, such as a refrigeration cycle. Assuming a refrigeration cycle instead of the stream from `raw-2` to `prod-3` in the example in Figure 3 yields the modified graph in Figure 7. The graph traversal starting from the virtual node first explores the sub-graph containing the distillation system and results in

```
(raw)(hex){1}(dist)[{bout}(prod)]{tout}(hex){1}(prod).
```

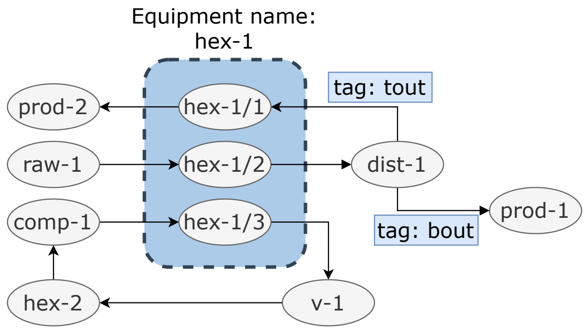


Figure 7: PFD graph with refrigeration cycle as sub-graph

Since the nodes of the refrigeration cycle are still not visited, we need another DFS in this sub-graph. Because there is no inlet node in the refrigeration cycle, the node with the lowest rank which is not an outlet node (out-degree=0), in this case, `hex-1/3`, is connected to the virtual node and selected as the initial node. The final SFILES 2.0 string is

```
(raw)(hex){1}(dist)[{bout}(prod)]{tout}(hex){1}(prod)n|(hex)<1(comp)(hex){1}(v)1.
```

4.3 Conversion from SFILES 2.0 string to flowsheet graph

string --> graph :
string을 notation rule에 따라 하나씩 노드와 엣지를 추가해가며 그래프 생성

The conversion of the SFILES 2.0 string back to a flowsheet graph is done by traversing the string and adding the nodes and edges according to the SFILES 2.0 notation rules. Note that the node numbering happens before the string traversal and is according to the order of occurrence in the SFILES 2.0 string. The latter implies that the node numbers of the original flowsheet graph and the reconstructed version might differ. However, the topology of the translated flowsheet information is preserved.

5 Conclusions

This paper is a proposition of the SFILES 2.0, containing modifications and extensions of the previously used SFILES. The development aims to include all essential topological information of flowsheets in the SFILES representation, such as a distinction between top- and bottom branches of unit operations. Moreover, the SFILES 2.0 includes a concept to describe control structures, which are mandatory for the operation of chemical plants. This extends the applicability of SFILES 2.0 from PFDs to P&IDs, which are the predominant diagram types utilized during the development and operation of chemical plants. To leverage the full potential regarding future databases, the SFILES 2.0 notation comes with naming conventions for the unit operations and a set of standardized stream tags. Eventually, the implementation of the reversible conversion between flowsheet graph and SFILES 2.0 strings is openly accessible to enable researchers and engineers to write or read SFILES 2.0 strings. This work attempts to lay the foundation for creating an SFILES 2.0-based database for PFDs and P&IDs, ideally containing a large variety of chemical processes.

6 Acknowledgements

This publication is part of the project "ChemEng KG – The Chemical Engineering Knowledge Graph" with project number 203.001.107 of the research programme "Open Science (OS) Fund 2020/2021" which is (partly) financed by the Dutch Research Council (NWO).

A Appendix

We consider the flowsheet in Figure 8. The SFILES 2.0 string for this flowsheet is:

`(raw)(pp)(r)<&|(raw)(mix)<1(dist)[{tout}(hex)&]{bout}(splt)1(prod)|(hex)(prod).`

The process branch that converges into the reactor is marked light blue in the SFILES 2.0 string and in the figure. According to the notation rules it is surrounded by `<&|` and `|` (highlighted in dark blue). The additional `&` sign indicates which node of the purple branch is connected to the reactor.

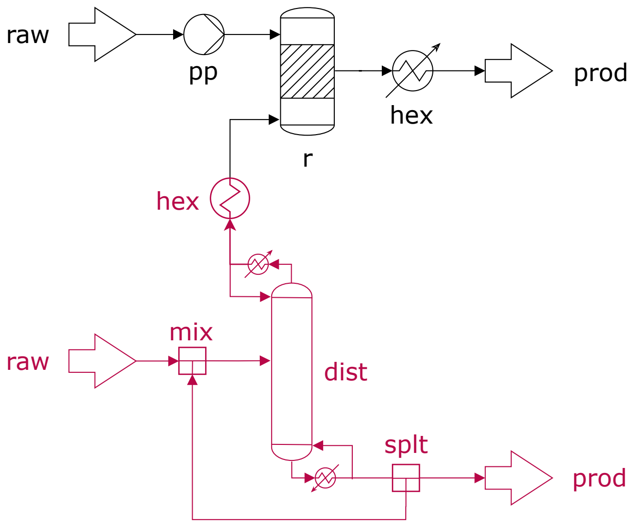


Figure 8: Flowsheet with multiple branchings

References

[1] Loïc d'Anterroches. *Process Flow Sheet Generation & Design through a Group Contribution Approach*. PhD thesis, Technical University of Denmark, March 2006.

[2] International Organization for Standardization. *Specifications for diagrams for process industry - Part 1: General rules*. ISO, 2010.

[3] International Organization for Standardization. *Specifications for diagrams for process industry - Part 2: Measurement and control*. ISO, 2015.

[4] Gavin P. Towler and R. K. Sinnott. *Chemical engineering design - Principles, practice and economics of plant and process design*. Elsevier/Butterworth-Heinemann, Amsterdam and Boston, 2008. ISBN 9780750684231.

[5] Moe Toghraei. *Piping and Instrumentation Diagram Development*. John Wiley & Sons, 2019. ISBN 9781119329343. URL `https://www.ebook.de/de/product/36019424/moe_toghraei_piping_and_instrumentation_diagram_development.html`.

[6] Michael Wiedau, Lars von Wedel, Heiner Temmen, Richard Welke, and Nikolaos Papakonstantinou. ENPRO data integration: Extending DEXPI towards the asset lifecycle. *Chemie Ingenieur Technik*, 91(3):240–255, jan 2019. doi:10.1002/cite.201800112.

[7] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers,

Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1), mar 2016. doi:10.1038/sdata.2016.18.

[8] Artur M. Schweidtmann, Erik Esche, Asja Fischer, Marius Kloft, Jens-Uwe Repke, Sebastian Sager, and Alexander Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Technik*, 93(12):2029–2039, oct 2021. doi:10.1002/cite.202100083.

[9] Tong Zhang, Nikolaos V. Sahinidis, and Jeffrey J. Siirola. Pattern recognition in chemical process flowsheets. *AIChE Journal*, 65(2):592–603, nov 2018. doi:10.1002/aic.16443.

[10] Chenglin Zheng, Xi Chen, Tong Zhang, Nikolaos V. Sahinidis, and Jeffrey J. Siirola. Learning process patterns via multiple sequence alignment. *Computers & Chemical Engineering*, page 107676, jan 2022. doi:10.1016/j.compchemeng.2022.107676.

[11] David Weininger, Arthur Weininger, and Joseph L. Weininger. SMILES. 2. algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, may 1989. doi:10.1021/ci00062a008.

[12] Anjan K. Tula, Mario R. Eden, and Rafiqul Gani. ProCAFD: Computer-aided tool for sustainable process synthesis, intensification and hybrid solutions. In *Computer Aided Chemical Engineering*, pages 481–486. Elsevier, 2019. doi:10.1016/b978-0-12-818634-3.50081-3.

[13] Anjan K. Tula, Mario R. Eden, and Rafiqul Gani. Hybrid method and associated tools for synthesis of sustainable process flowsheets. *Computers & Chemical Engineering*, 131:106572, dec 2019. doi:10.1016/j.compchemeng.2019.106572.

[14] Gabriel Vogel, Lukas Schulze Balhorn, Edwin Hirtreiter, and Artur M. Schweidtmann. process-intelligence-research/sfiles2: v1.0.0, July 2022. URL `https://doi.org/10.5281/zenodo.6901932`.

[15] Hector Yeomans and Ignacio E. Grossmann. A systematic modeling framework of superstructure optimization in process synthesis. *Computers & Chemical Engineering*, 23(6):709–731, jun 1999. doi:10.1016/s0098-1354(99)00003-4.

[16] Henry Winter and Marina Böckelmann. *Prozessleittechnik in Chemieanlagen*, volume 5. Verlag Europa-Lehrmittel Nourney Vollmer, Haan-Gruiten, 2015. ISBN 9783808571002.

[17] Jan Morbach, Andreas Wiesner, and Wolfgang Marquardt. Ontocape—a (re)usable ontology for computer-aided process engineering. *Computers & Chemical Engineering*, 33(10):1546–1556, 2009. ISSN 0098-1354. doi:https://doi.org/10.1016/j.compchemeng.2009.01.019. URL `https://www.sciencedirect.com/science/article/pii/S0098135409000362`. Selected Papers from the 18th European Symposium on Computer Aided Process Engineering (ESCAPE-18).

[18] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, may 1965. doi:10.1021/c160017a018.