SFILES 2.0

an extended text-based flowsheet representation

Vogel, Gabriel; Hirtreiter, Edwin; Schulze Balhorn, Lukas; Schweidtmann, Artur M.

DOI
10.1007/s11081-023-09798-9

Publication date
2023

Document Version
Final published version

Published in
Optimization and Engineering

Citation (APA)
Vogel, G., Hirtreiter, E., Schulze Balhorn, L., & Schweidtmann, A. M. (2023). SFILES 2.0: an extended text-based flowsheet representation. *Optimization and Engineering*, *24*(4), 2911-2933. https://doi.org/10.1007/s11081-023-09798-9

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Optimization and Engineering
https://doi.org/10.1007/s11081-023-09798-9

RESEARCH ARTICLE

SFILES 2.0: an extended text-based flowsheet representation

Gabriel Vogel[1] · Edwin Hirtreiter[1] · Lukas Schulze Balhorn[1] · Artur M. Schweidtmann[1]

Received: 26 July 2022 / Revised: 21 March 2023 / Accepted: 22 March 2023

Abstract
SFILES are a text-based notation for chemical process flowsheets. They were originally proposed by d'Anterroches (Process flow sheet generation & design through a group contribution approach) 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

✉ Artur M. Schweidtmann
a.schweidtmann@tudelft.nl

[1] Department of Chemical Engineering, Delft University of Technology, Van der Maasweg 9, 2629 HZ Delft, The Netherlands

Published online: 02 May 2023

1 Introduction

Chemical process flowsheets, also known as process flow diagrams (PFDs) ISO (2010, 2015), are the current standard for depicting and communicating the topology of unit operations in chemical processes (see Fig. 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) ISO (2010, 2015) are a central representation class of chemical processes. They include additional information about instrumentation, valves, control structures, and piping Towler and Sinnott (2008). 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 Toghraei (2019). 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 Wiedau et al. (2019), 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) Wilkinson et al. (2016) 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 (Schweidtmann et al. 2021).

Chemical flowsheets can be represented as directed graphs (Zhang et al. 2018; Zheng et al. 2022). The flowsheet graph (see Fig. 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 (2006) 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) (Weininger et al. 1989) 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. (2019a, b) 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 Zhang et al. (2018; 2022). 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 Vogel et al. 2022 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.

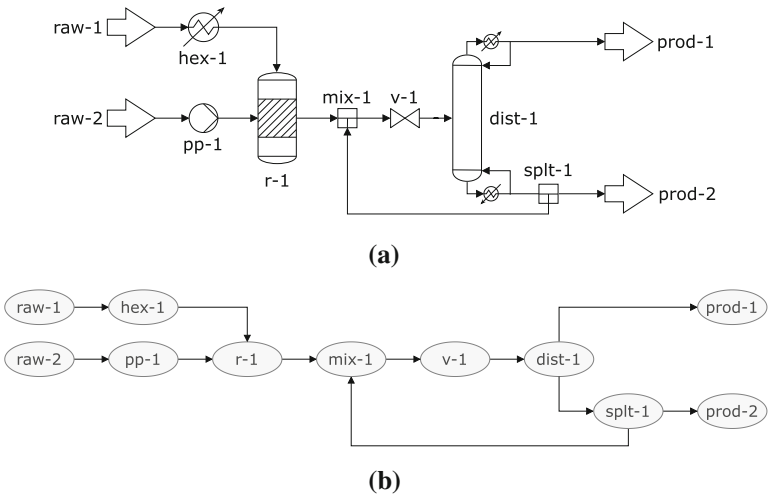


Fig. 1 **a** Simple chemical process flowsheet with branches and one recycle stream. **b** Graph representation of the flowsheet in (**a**)

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 (d'Anterroches 2006) 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 in Zhang et al. (2018) 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 Fig. 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 Fig. 1b `r-1, mix-1`), and a node with an out-degree>1 indicates a new branching at the considered node (in Fig. 1b `dist-1, splt-1`).

2.2 Original SFILES notation

The original SFILES notation rules (d'Anterroches 2006) are outlined in this section using the flowsheet graph in Fig. 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 Fig. 1a, all except the last considered branch during the conversion from flowsheet graph to string (see Sect. 4.1), are noted in square brackets. In the case of converging branches at a node with an in-degree>1, the original definition d'Anterroches (2006) 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 (d'Anterroches 2006):

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 Sect. 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 Zhang et al. (2018; 2022) 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 Fig. 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, Sect. 3.2 describes the notation details that are required to represent the control structure contained in P&IDs. Additionally, we propose standardized naming conventions for commonly used unit operations in Sect. 3.3. Finally, to enhance the usability for other researchers in the field, we created Tables 3 and 4 in Sect. 3.5 summarizing the SFILES 2.0 syntax and notation rules. In the following, we use generalized SFILES as the standard notation (see Sect. 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 Sect. 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 Fig. 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 the flowsheet in Fig. 2. Using the conversion algorithm described in Sect. 4.2 the SFILES 2.0 string for this flowsheet is:

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

The process branch that converges into the reactor is marked dark red 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 dark red branch is connected to the reactor.

Furthermore, 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

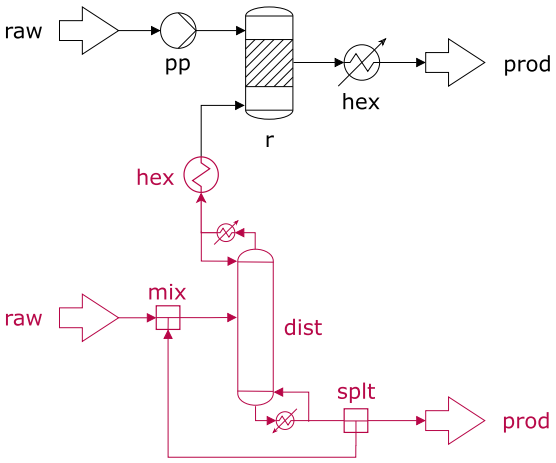


Fig. 2 Flowsheet with multiple branchings.The branch in dark red colour is a converging branch in the SFILES 2.0 string consisting of a branching itself. It illustrates the necessity of our modification of adding a & sign to encode how the branches are connected

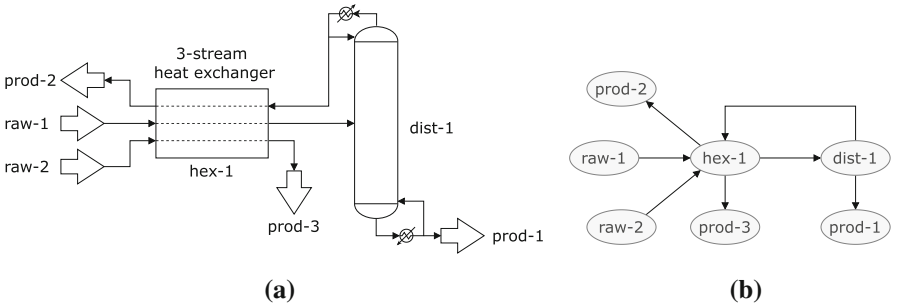


Fig. 3 Flowsheet with complex connectivity characteristics. **a** PFD, **b** graph representation

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 Fig. 3 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.

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) (Zhang et al. 2018; Yeomans and Grossmann 1999) for the representation of the superstructure of chemical processes.

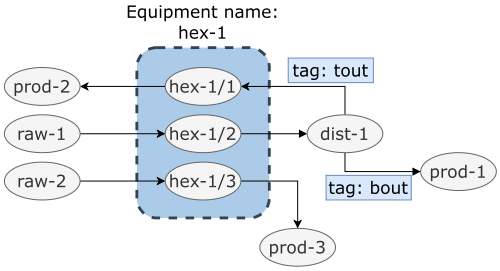


Fig. 4 Flowsheet graph with modified node structure of heat exchanger and connectivity attributes for distillation column

As shown in Fig. 4, 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. The authors in Zhang et al. (2018) 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)`

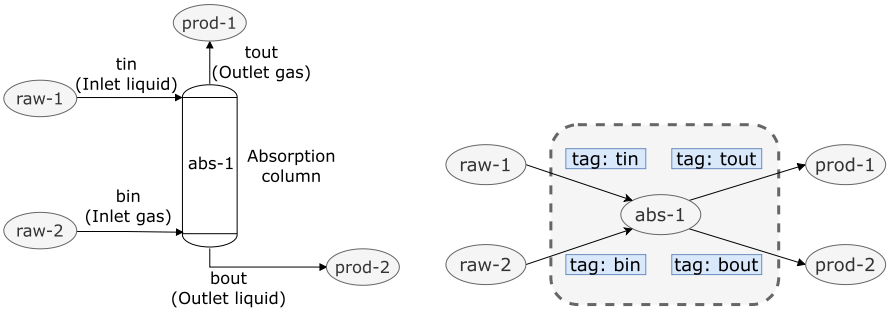


Fig. 5 **a** Absorption column with two inlets and two outlets. **b** Flowsheet graph of **a** with connectivity stream tags

3. `(v)<1...(u){x}1`
4. `(v)<&|...(u){x}&|`

The stream tags must be saved as edge attributes in the flowsheet graph, e.g., the top and bottom outlet stream tags in Fig. 4. 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 Fig. 3. The stream tags can also be applied to other unit operations such as absorption or extraction columns. Figure 5a 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 Fig. 5b. The SFILES 2.0 string for this hypothetical process is

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

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.

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 Fig. 6. The first example (i) in Fig. 6a consists of a sensor measuring the flow rate of a stream and controlling the subsequent valve with this information. Since material streams are implicitly

Table 1 Set of stream and control tags used in SFILES 2.0. A complete list of possible control tags according to DIN EN 62424 can be found in Winter and Böckelmann (2015)

Connectivity information	Stream tag in flowsheet graph and SFILES 2.0
Bottom inlet	bin
Top inlet	tin
Bottom outlet	bout
Top outlet	tout
Multi-stream heat exchanger identifier	# (unique number per heat exchanger)
Control structure information	**Control tag in flowsheet graph and SFILES 2.0**
Flow control	FC
Level control	LC
Pressure control	PC
Temperature control	TC

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) Winter and Böckelmann (2015), 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 Fig. 6a:

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

The second example (ii) in Fig. 6b 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 Fig. 6c 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 Fig. 6c:

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

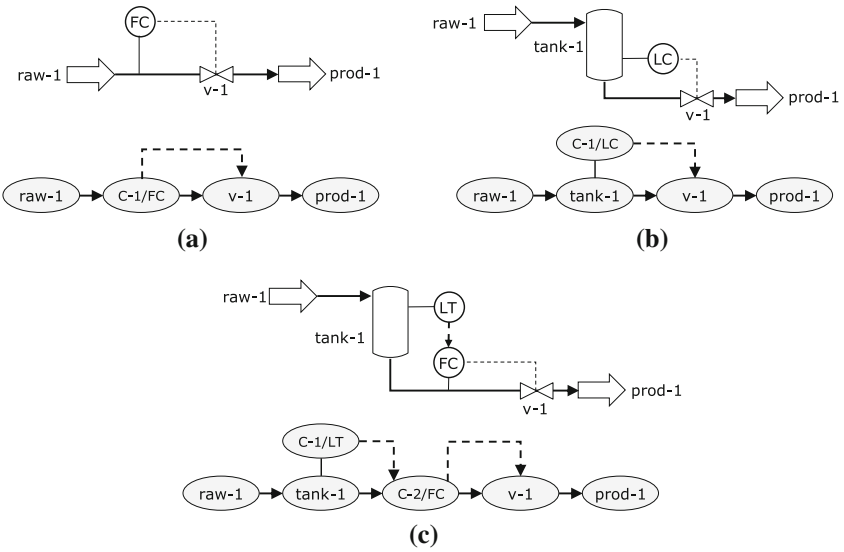


Fig. 6 PFD and flowsheet graph of simple control loops. **a** Flow control of material stream, **b** Level control of tank, **c** Level control of tank with control cascade

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 (Morbach et al. 2009). 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.

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.

Table 2 Unit operations and abbreviations in SFILES 2.0 based on OntoCAPE ontology (Morbach et al. 2009)

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

Table 2 continued

Unit operation	Abbreviation	OntoCAPE term	Typical in-degree	Typical out-degree
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	X	-	-	-

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

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.

3.5 Summary of SFILES 2.0 rules

This subsection provides a summary of the SFILES 2.0 rules, i.e., Table 3 summarizes the general rules, whereas Table 4 shows the notation rules specifically defined for P&IDs.

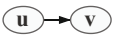
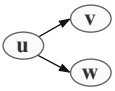
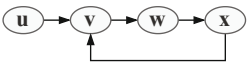
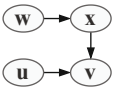
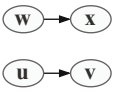
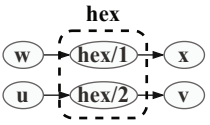
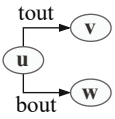
4 SFILES 2.0 generation algorithm

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 Vogel et al. 2022. Similar to the original SFILES notation algorithm (see Sect. 2.2), the two major steps for the SFILES 2.0 string generation are the determination of the graph invariant (Sect. 4.1) and the graph traversal (Sect. 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 Sect. 3.2.

4.1 Determination of graph invariant

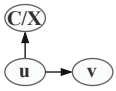
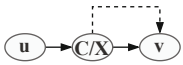
The graph invariant aims to yield a unique rank for each node. The determination of this graph invariant is also known as graph canonization. The first step in our implementation is based on the Morgan algorithm (Morgan 1965), similarly to the

Table 3 Summary of SFILES 2.0 rules: PFD and P&ID related

SFILES 2.0 sequence	Simplified graph	Rule explanation
(u)(v)		Two subsequent unit operations in parenthesis (u) and (v) imply a directed connection from (u) to (v)
(u)[(v)](w)		Square brackets indicate a branching. The unit operation (u) has two outlet connections to (v) and (w). Every branch except the last one (here, (w) is not in brackets) is noted in brackets.
(u)(v)<1(w)(x)1		The symbols <# and # (# is a number) indicate a recycle connection. In this case the recycle connection is from (x) to (v).
(w)(x)(v)<&|(u)&|		Incoming new branches are denoted between <&| and &|. In this example the incoming branch at (v) starts from a new inlet node (w); the & sign indicates the connection from (x) to (v).
(u)(v)n|(w)(x)		The n| indicates a new separate process (w)(x) with no material stream connection to the previous process (u)(v).
(u)(hex){1}(v) n|(w)(hex){1}(x)		The notation # after a heat exchanger unit is used for multi-stream heat exchangers to identify which streams share a specific heat exchanger (heat integration).
(u)[{tout}(v)]{bout}(w)		Stream tags are noted in braces, e.g. {tout} for top outlet, which is used to store information for unit operations with multiple in-/outlet streams, where the position of in-/outlet is important for the process description (e.g. distillation, absorption,...).

description in Zhang et al. (2018). As illustrated in Fig. 7, 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

Table 4 Summary of SFILES 2.0 rules: only P&ID related

SFILES 2.0 sequence	Simplified graph	Rule explanation
(u)(C){X}(v)		Control units C for measuring the state of a stream, e.g. a flow indicator (FI), are noted between the two unit operations u and v where the measurement takes place. The letter code X is noted in braces directly after the control unit C, which is noted in parenthesis.
(u)[(C){X}](v)		The control unit C with the letter code X noted in square brackets indicate that the control unit, e.g. a level control (LC), is directly connected to a unit operation (here, u).
(u)(C){X}_1(v)<_1		Signal connections between control units or control units and unit operations are indicated with <_# and _#. The smaller than sign defines the direction: here, the control unit C with the letter code X is connected to the unit operation v.

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 Sect. 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.

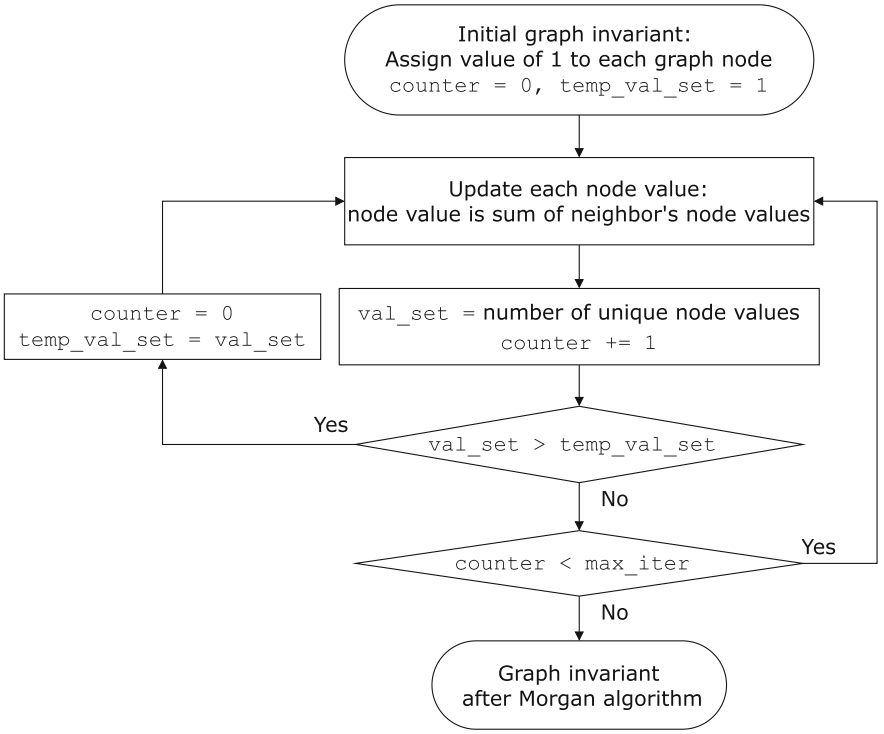


Fig. 7 Morgan algorithm for graph invariant determination

Table 5 Node ranks for flowsheet graph in Fig. 1b

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

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 5 shows the node ranking for the example in Fig. 1b.

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 Fig. 1b, we will explain how the DFS algorithm and the SFILES string generation work. According to Fig. 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 5, 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 Fig. 4 yields the modified graph in Fig. 8. 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).
```

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

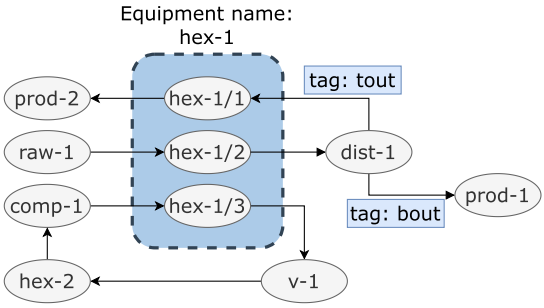


Fig. 8 PFD graph with refrigeration cycle as sub-graph

```
(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

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 Illustrative examples

This Section provides additional examples of flowsheets with a higher number of unit operations and control structures. Figure 9 shows the process flow diagram for the production of maleic anhydride from benzene which was extracted from a DWSIM simulation file. The corresponding flowsheet graph contains 22 nodes. Converting the flowsheet graph to the SFILES 2.0 representation yields:

```
(raw)(pp)(hex){1}(mix)<&|(raw)(mix)<&|(raw)&|(comp)&|(hex)
    (mix)<1(r)[(hex)1](hex)(mix)<2<&|(raw)&|(sep)[{tout}
    (prod)]{bout}(dist){bout}2{tout}(prod)n|(raw)(hex)
    {1}(prod).
```

Fig. 10 shows the PFD of a natural gas processing unit with many branches. The corresponding SFILES 2.0 string is:

```
(raw)(hex)(flash)[{bout}(prod)]{tout}(hex)(flash)[{tout}
    (turb)(flash)[{bout}(hex)(mix)<2(hex)(dist)[{bout}
    (dist)[{tout}(prod)]{bout}(dist)[{bout}(prod)]{tout}
    (prod)]{tout}(mix)<1(hex)(comp)(comp)(prod)]{tout}
    (hex)1]{bout}(hex)2
```

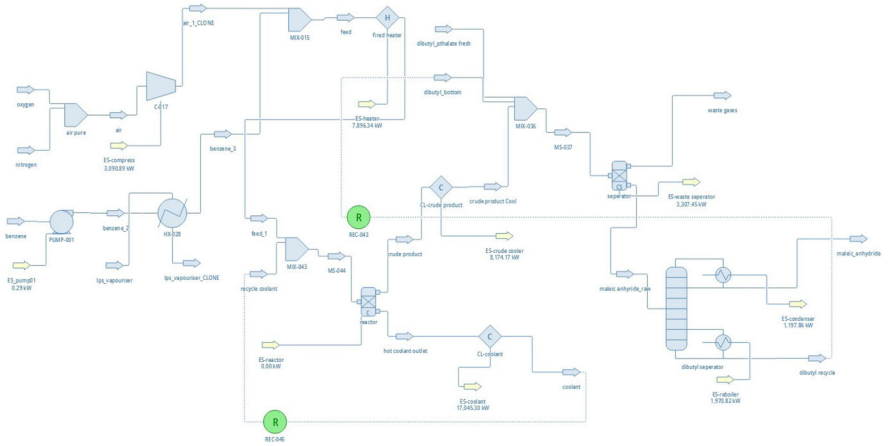


Fig. 9 Process flow diagram for maleic anhydride production from benzene. (Badodekar n.d.). CC BY-SA 4.0

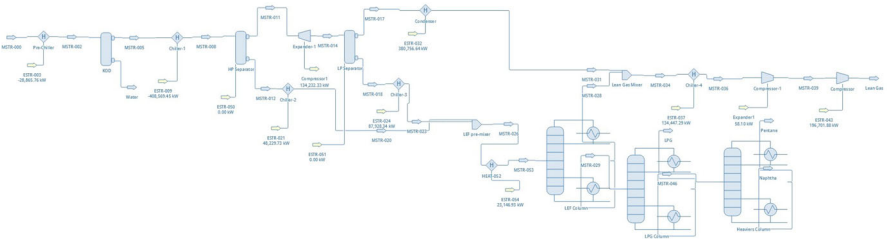


Fig. 10 Process flow diagram of a natural gas processing unit. (Shah et al. 2018). CC BY-SA 4.0

Fig. 11 shows a P&ID of a distillation column with a high number of unit operations and control structures. The corresponding SFILES 2.0 string is:

```
(raw)(C){FC}_1(v)<_1(mix)<&|(raw)(C){FC}_2(v)&<_2|(hex)
    {1}(C){TC}_3(dist)<1<2[(C){PC}_4][(C){LC}_5][{tout}
    (hex)(sep)[(C){LC}_6][(v)<_4(prod)](splt)[(C){FC}_7
    (v)<_7(prod)](v)1<_6]{bout}(splt)[(v)<_5(prod)](hex)
    {2}2n|(raw)(splt)[(hex){1}(mix)<3(prod)](v)3<_3n|(raw)
    (C){FC}_8(v)<_8(hex){2}(prod).
```

Fig. 12 shows the P&ID of a two-stage flash process with control structures. The corresponding SFILES 2.0 string is:

```
(raw)(C){FC}_1(v)<_1(hex){1}(C){TC}_2(sep)[(C){PC}_3]
    [(C){LC}_4][(v)<_3(prod)](C){FC}_5<_4(v)<_5(sep)
    [(C){PC}_6][(C){LC}_7][(v)<_6(prod)](C){FC}_8<_7(v)
    <_8(prod)n|(raw)(v)<_2(hex){1}(prod).
```

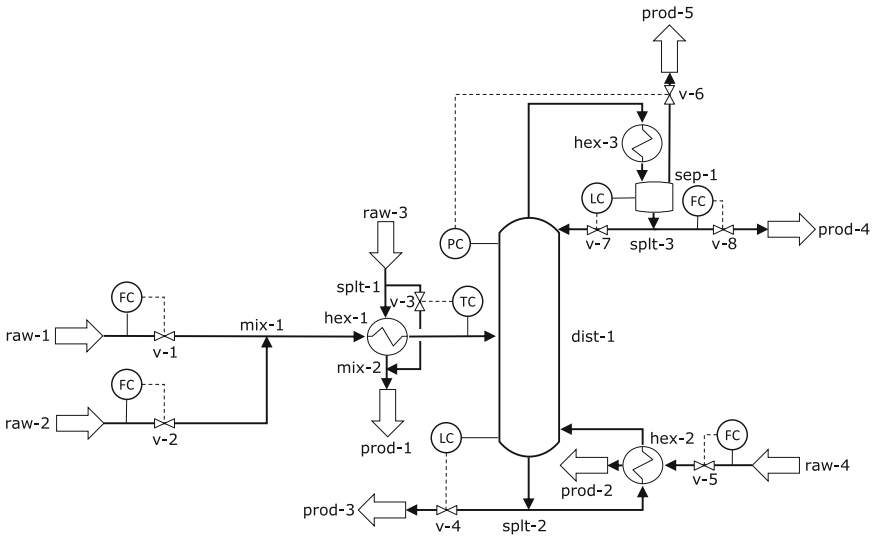


Fig. 11 Process flowsheet of a distillation column with control structure

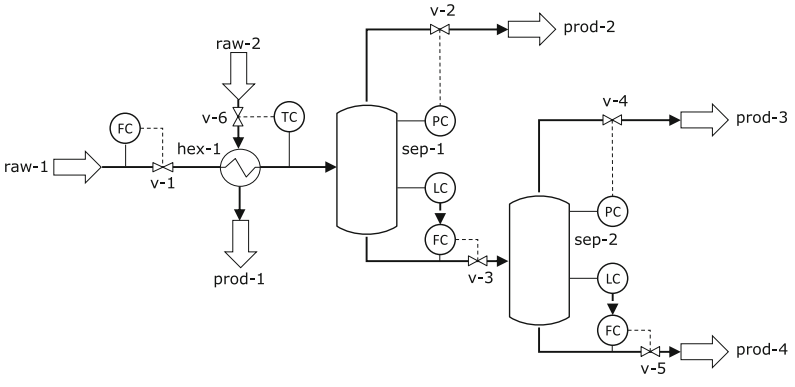


Fig. 12 Two-stage flash process flowsheet with control structure

6 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 con-

version 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.

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).

References

Badodekar S (n.d.) Production of Maleic Anhydride from benzene. DWSIM. https://dwsim.fossee.in/flowsheeting-project/dwsim-flowsheet-run/83

d'Anterroches L (March 2006) Process flow sheet generation & design through a group contribution approach. PhD thesis, Technical University of Denmark

for Standardization IO (2010) Specifications for diagrams for process industry - Part 1: general rules. ISO, Geneva, Switzerland

for Standardization IO (2015) Specifications for diagrams for process industry - Part 2: measurement and control. ISO, Geneva, Switzerland

Morbach J, Wiesner A, Marquardt W (2009) Ontocape-a (re)usable ontology for computer-aided process engineering. Comput Chem Eng 33(10):1546–1556. https://doi.org/10.1016/j.compchemeng.2009.01.019. (Selected Papers from the 18th European Symposium on Computer Aided Process Engineering (ESCAPE-18))

Morgan HL (1965) The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. J Chem Document 5(2):107–113. https://doi.org/10.1021/c160017a018

Schweidtmann AM, Esche E, Fischer A, Kloft M, Repke J-U, Sager S, Mitsos A (2021) Machine learning in chemical engineering: a perspective. Chemie Ingenieur Technik 93(12):2029–2039. https://doi.org/10.1002/cite.202100083

Shah D, Hemanth R, Aditya D (2018) Natural Gas Processing Simulation. DWSIM. https://dwsim.fossee.in/flowsheeting-project/dwsim-flowsheet-run/122

Toghraei M (2019) Piping and instrumentation diagram development. John Wiley & Sons, Hoboken, New Jersey, USA. https://www.ebook.de/de/product/36019424/moe_toghraei_piping_and_instrumentation_diagram_development

Towler GP, Sinnott RK (2008) Chemical engineering design - principles. Practice and Economics of Plant and Process Design. Elsevier/Butterworth-Heinemann, Amsterdam and Boston

Tula AK, Eden MR, Gani R (2019a) ProCAFD: Computer-aided tool for sustainable process synthesis, intensification and hybrid solutions. In: computer aided chemical engineering, pp 481–486. Elsevier, Amsterdam, Netherlands. https://doi.org/10.1016/b978-0-12-818634-3.50081-3

Tula AK, Eden MR, Gani R (2019b) Hybrid method and associated tools for synthesis of sustainable process flowsheets. Comput Chem Eng 131:106572. https://doi.org/10.1016/j.compchemeng.2019.106572

Vogel G, Balhorn LS, Hirtreiter E, Schweidtmann AM (2022) Process-intelligence-research/SFILES2: V1.0.0. https://doi.org/10.5281/zenodo.6901932

Weininger D, Weininger A, Weininger JL (1989) SMILES. 2. algorithm for generation of unique SMILES notation. J Chem Inf Comput Sci 29(2):97–101. https://doi.org/10.1021/ci00062a008

Wiedau M, von Wedel L, Temmen H, Welke R, Papakonstantinou N (2019) ENPRO data integration: extending DEXPI towards the asset lifecycle. Chemie Ingenieur Technik 91(3):240–255. https://doi.org/10.1002/cite.201800112

Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten J-W, da Silva Santos LB, Bourne PE, Bouwman J, Brookes AJ, Clark T, Crosas M, Dillo I, Dumon O, Edmunds S, Evelo CT, Finkers R, Gonzalez-Beltran A, Gray AJG, Groth P, Goble C, Grethe JS, Heringa J, t' Hoen PAC, Hooft R, Kuhn T, Kok R, Kok J, Lusher SJ, Martone ME, Mons A, Packer AL, Persson B, Rocca-Serra P, Roos M, van Schaik R, Sansone S-A, Schultes E, Sengstag T, Slater T, Strawn G, Swertz MA, Thompson M, van der Lei J, van Mulligen E, Velterop J, Waagmeester A, Wittenburg P, Wolstencroft K, Zhao J, Mons B, (2016) The FAIR guiding principles for scientific data management and stewardship. Sci Data. https://doi.org/10.1038/sdata.2016.18

Winter H, Böckelmann M (2015) Prozessleittechnik in chemieanlagen, vol 5. Verlag Europa-Lehrmittel Nourney Vollmer, Haan-Gruiten

Yeomans H, Grossmann IE (1999) A systematic modeling framework of superstructure optimization in process synthesis. Comput Chem Eng 23(6):709–731. https://doi.org/10.1016/s0098-1354(99)00003-4

Zhang T, Sahinidis NV, Siirola JJ (2018) Pattern recognition in chemical process flowsheets. AIChE J. 65(2):592–603. https://doi.org/10.1002/aic.16443

Zheng C, Chen X, Zhang T, Sahinidis NV, Siirola JJ (2022) Learning process patterns via multiple sequence alignment. Comput Chem Eng. https://doi.org/10.1016/j.compchemeng.2022.107676