Steady-state chemical process models. A structural point of view

Ali Baharev, Kevin Kofler, Arnold Neumaier

April 30, 2013

Contents

1 Introduction **4**

1.1 Limitations 4

2 Process streams **5**

3 Sources and sinks **5**

4 Atomic units **6**

4.1 Structural types of the atomic units 6

5 Equations **7**

5.1 Type of equations 7

5.2 Balance equations 7

5.3 Mechanical equilibrium 9

5.4 Thermal equilibrium 9

5.5 Phase equilibrium 9

5.6 Degrees of freedom 11

6 Model equations of the atomic units **11**

6.1 Divider 11
6.2 Mixer 12
6.2.1 Heat of mixing 13
6.3 Heat exchanger 14
6.4 Pressure changer 14
6.5 Reactor 15
6.6 Flash 15
6.7 Reactive VLE flash 16
6.8 Partial reboiler 18

7 Composite units **19**

7.1 Interpretation of the connecting equations in the implementation 19
7.2 VLE stage 20
7.3 VLE cascade 21
7.4 Total condenser 22
7.5 Total reboiler 24
7.6 A practical example 24

8 Flash calculations **26**

8.1 Notes 27
8.2 Degenerate cases 27
8.3 A practical example of the phase equilibrium condition 27

9 Activity coefficient models **28**

9.1 Wilson model 28
9.2 NRTL 28
9.3 Extended NRTL (a particular one) 28
9.4 UNIQUAC (a particular formulation) 29

10 Separation operations **29**

10.1 Internal physical structure of distillation columns 30

10.2 Example: multiple steady-states in ideal two-product distillation 30

11 Test examples **31**

11.1 Reactive distillation column for manufacturing ethylene glycol . . 31

11.1.1 Model description 32

11.1.2 Elimination order 33

11.2 Column of Jacobsen and Skogestad 35

12 Appendix **36**

1 Introduction

Chemical processes are represented by connected, directed graphs. A node that is neither a source nor a sink is called a **unit**. If a unit can be unfolded then it is a **composite unit**. The unfolding procedure can be repeated recursively until no composite unit remains, only **atomic units**.

The directed edges are called **process streams**. Each edge corresponds to its own set of variables. With the exception of one of the stream variables, all of them have non-negativity bound constraints. The direction of the edges correspond to the material flow direction in the chemical process.

The input and output streams of a node are referred to as **inlets** and **outlets**, respectively, and their variables as **in-variables** and **out-variables**, respectively. The nodes relate variables of different process streams by enforcing joint constraints on the in- and out-variables.

Units (nodes) can have additional **internal variables**. Some of these variables have special significance, they are called **unit parameters**. The unit parameters play an import role in designing chemical processes and are often design variables in process optimization.

Local specifications are additional constraints for variables that belong to a single unit and / or to streams associated with a single unit. Usually, these specifications correspond to a closed-loop control system. The form of the specification equations shows large variation: they can be trivial equations fixing the value of a variable as well as complicated nonlinear equations.

Seldom used specifications are the **non-local specifications**: These inequality constraints reference variables from non-adjacent streams or variables from more than one unit. This extra information usually comes from engineering considerations or from intuition. For example, non-local specifications can reflect information on what is considered a practically meaningful operation by the engineer. Another reason to use non-local specifications is to improve performance by passing additional information to the underlying solver.

Code snippets, showing the implementation of the units, will be given near there mathematical model of the unit. The implementation has been carried out in the Concise environment, see F. Domes [5] and P. Schodl [8]. The entire implementation and the Concise typesheet is given in the Appendix.

1.1 Limitations

This work is **not** aiming at modeling of multi-domain, complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents. Only flows of chemicals are considered in the current work. Heat flows allowing thermal coupling or multi-domain models would need an extension.

It is assumed that the kinetic term, the potential term and the friction losses can be neglected in the heat balance as it is customery in chemical process modeling. (If, for some reason, these terms have to be considered, one can compensate for them by introducing pressure changers and heat exchangers appropriately and enforcing specifications that account for these neglected terms.)

The thermodynamically consistent model of the mixer (see Section 6.2) is nonlinear. However, it is conventionally treated as linear in the chemical engineering literature by neglecting the so-called heat of mixing, see Subsection 6.2.1. In the current work, the heat of mixing is also neglected: A nonlinear mixer has a domino effect, many of the composite units would be no longer worth decomposing.

2 Process streams

A process **stream** S, consisting of C **components**, is characterized by the list

$$S = \{S.f_{1:C},\ S.p,\ S.H\}$$

of $C + 2$ independent variables, see also Table 1. To identify the stream S to which a variable x belongs, we write it as $S.x$. The implementation is given in Listing 1. Process streams are represented by arrows as shown in Figure 1.

variable	physical meaning	SI unit
$S.f_i \geq 0$	molar flow rate of component $i = 1 : C$	mol/s
$S.p \geq 0$	pressure	Pa
$S.H$	enthalpy flow rate	J/s

Table 1: The $C + 2$ independent variables characterizing the process stream S.

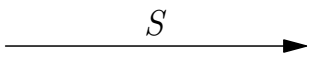

Figure 1: The graphical representation of stream S.

3 Sources and sinks

The input of the chemical process is provided by the **sources** and its output is consumed by the **sinks**. A sources has exactly one outlet; a sink has exactly one inlet. In the modeling language, sources and sinks are reserved keywords.

Listing 1: Implementation of the process stream.

```
parameters {
  C .. natural number   % number of chemical substances
}

quantities {
  molar flow rate (mol/s) >= 0
  pressure (Pa) >= 0
  enthalpy flow rate (J/s)
}

stream {
  f[C] .. molar flow rate
  p .. pressure
  H .. enthalpy flow rate
}
```

The **null sink** is used to express the absence of a stream; it enforces the constraint

$$\sum A.f_i = 0$$

on its inlet A.

4 Atomic units

These units are the followings.

1. Mixer
2. Heat exchanger
3. Pressure changer
4. Reactor
5. Divider
6. Flash
7. Reactive flash
8. Partial reboiler

4.1 Structural types of the atomic units

The mixer has multiple inlets and a single outlet. All other atomic units have a single inlet and can have either one or two outlets. See Figure 2.

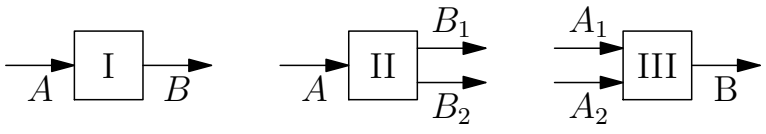

Figure 2: Structural types of the atomic units: (I) heat exchanger, pressure changer, reactor; (II) divider, flash, partial reboiler; (III) mixer.

5 Equations

A generic unit is described by the collection $A = \{A_1, A_2, \ldots, A_n\}$ of its inlets A_j, the collection $B = \{B_1, B_2, \ldots, B_m\}$ of its outlets B_k, the collection $v = \{v_1, v_2, \ldots, v_\ell\}$ of its internal variables and unit parameters v_p, and appropriate equations $E(A, B, v) = 0$ and inequality constraints on some of the internal variables. The graphical representation of a generic unit is shown in Figure 3.

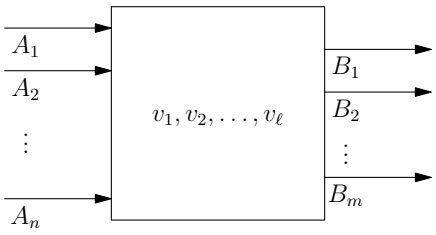

Figure 3: Graphical representation of a generic unit with input streams A_j, output streams B_k, and internal variables v_p.

5.1 Type of equations

Material balances: A system of C linear equations, reflecting the conservation of mass.

Heat balance: A linear equation reflecting the conservation of energy.

Mechanical equilibrium: The outlets have the same pressure as the unit. With the exception of the mixer and the pressure changer, the pressure of the unit equals the pressure of its only inlet and the pressure is not considered as an internal variable.

Thermal equilibrium: The outlets have the same temperature as the unit. If the temperature is not an internal variable of the unit then these equations are missing.

Phase equilibrium: $C \times (P - 1)$ (P denotes the number of phases) nonlinear equations expressing the fact that the **chemical potential** of any of the components is the same in all phases. These equations apply only to the flash units (flash, reactive VLE flash, partial reboiler).

Characterizing equations: These equations characterize how the unit works and cannot be changed.

5.2 Balance equations

The balance equations must hold for each unit (both atomic and composite units). Altogether, there are $C + 1$ balance equations, C material balances and

the heat balance.

1. **Chemical reactions.** We assume that the components $M_1, M_2, \ldots, M_C$ take part in R reactions,

$$\sum_{i=1}^{C} \nu_{ir} M_i = 0, \quad (r = 1 : R),$$

with fixed **stoichiometric coefficients** ν_{ir} whose sign is given by

$$\nu_{ir} \begin{cases} < 0 & \text{if component } i \text{ is a product in reaction } r, \\ = 0 & \text{if component } i \text{ is inert in reaction } r, \\ > 0 & \text{if component } i \text{ is a reactant in reaction } r. \end{cases}$$

There is one internal variable associated with each chemical reaction r, the **extent of reaction** $\xi_r \geq 0$. Additionally, each reaction is associated with a constant, the **heat of reaction** ΔH_r which specific to the reaction. Hence, correspondingly many equations are needed if the steady-state model of the unit needs to be properly specified. These equations can be as simple as fixing ξ_r to constant values. However, in general, the **rate equations**, determining the extent of reaction, are typically complicated nonlinear functions of the temperature and other variables.
Reactions only happen inside the reactor and the reactive flash atomic units. For all other atomic units $\xi_r = 0$ and $\Delta H_r = 0$.

2. The **material balance equations** are given by

$$\sum_{j=1}^{n} A_j.f_i = \sum_{k=1}^{m} B_k.f_i + \sum_{r=1}^{R} \nu_{ir}\xi_r \quad (i = 1 : C).$$

In particular, in the absence of reactions,

$$\sum_{j=1}^{n} A_j.f_i = \sum_{k=1}^{m} B_k.f_i \quad (i = 1 : C).$$

3. The **heat balance equation**

$$\sum_{j=1}^{n} A_j.H = \sum_{k=1}^{m} B_k.H + \sum_{r=1}^{R} \Delta H_r \xi_r + Q$$

involves the variable Q, called the **heat duty**; except for the heat exchanger and the partial reboiler, $Q = 0$ for all atomic units. In case of the heat exchanger and the partial reboiler, Q is either involved in a specification or its value is sought. It is only noted here that the so-called heat of mixing is conventionally neglected, see Subsection 6.2.1 for more details.

In the absence of reactions and with the exception of the heat exchanger and the partial reboiler, the heat balance equations take the following form:

$$\sum_{j=1}^{n} A_j.H = \sum_{k=1}^{m} B_k.H.$$

5.3 Mechanical equilibrium

In mechanical equilibrium, the outlets have the same pressure p as the atomic unit,

$$B_k.p = p \quad (k = 1 : m), \tag{1}$$

and the pressure p of an atomic unit is determined as

$$p = \min(A_j.p) + \Delta p.$$

Here, Δp is the pressure change associated with the unit; $\Delta p = 0$ for all atomic units except for the pressure changer.
Only the mixer has multiple inlets, that is, with the exception of the mixer, $\min(A_j.p)$ simplifies to $A.p$. In short, except for the mixer and the pressure changer, the pressure of an atomic unit equals the pressure of its only inlet.

5.4 Thermal equilibrium

The outlets have the same temperature T (and the same pressure p, see 1) as the atomic unit,

$$EOS(B_k, T) = 0 \quad (k = 1 : m),$$

where the equation of state EOS is chosen by the modeler. If the temperature is not an internal variable of the unit then these equations are missing.

5.5 Phase equilibrium

These equations apply only to the flash units (flash, reactive VLE flash, partial reboiler). In phase equilibrium, the chemical potential of any of the components is the same in all phases. Assuming that the atomic unit has two outlets B_1 and B_2, and there are exactly two phases inside the unit, the phase equilibrium condition has the following form

$$EOS_\ell(B_1.f, B_2.f, p, T, u) = 0 \quad (\ell = 1 : C + A), \tag{2}$$

where EOS is a suitable equation of state; p and T are the pressure and temperature of the atomic unit; u is the vector of auxiliary variables with dimension $A \geq 0$. Practical examples of Equation (2) are given in Section 8 and 9.

Listing 2: Implementation of the atomic unit base class.

```
parameters {
  C .. natural number   % number of chemical substances
}

quantities {
  molar flow rate (mol/s) >= 0
  pressure (Pa) >= 0
  enthalpy flow rate (J/s)
  reaction heat rate (J/s)
}

stream: material {
  f[C] .. molar flow rate
  p .. pressure
  H .. enthalpy flow rate
}

% the unnamed atomic unit carries the base equations
atomic unit {
  variables {
    reactionRate[C] .. molar flowrate
    reactionHeat .. reaction heat rate
    exchangedHeat .. enthalpy flow rate
    pressureDifference .. pressure

    p .. pressure
  }

  % named equation sets can be overridden
  equations: rate equations {
    reactionRate[1:C] = 0
    reactionHeat = 0
  }

  equations: exchanged heat {
    exchangedHeat = 0
  }

  equations: pressure change {
    pressureDifference = 0
  }

  % the unnamed equation set is invariant and can only be added to
  equations {
    for i in 1:C {
       sum(inlets[j].f[i] for j in 1:nInlets) = ...
         sum(outlets[k].f[i] for k in 1:nOutlets) + reactionRate[i]
    }

    sum(inlets[j].H for j in 1:nInlets) = ...
      sum(outlets[k].H for k in 1:nOutlets) + reactionHeat+exchangedHeat

    p = min(inlets[j].p for j in 1:nInlets) + pressureDifference
    for k in 1:nOutlets {
      outlets[k].p = p
    }
  }
}
```

5.6 Degrees of freedom

The $C+1$ balance equations, the thermal and mechanical equilibrium conditions on the outlets and the characterizing equations leave the unit underdetermined by n_{dof} **degrees of freedom**:

$$
\begin{aligned}
&n_{\mathrm{dof}} = n_{\mathrm{var}} - n_{\mathrm{eq}}, \quad \text{where} \\
&n_{\mathrm{var}} = (n+m)(C+2) + \ell \quad \text{and} \\
&n_{\mathrm{eq}} = (C+1) + m + d + t.
\end{aligned}
$$

The parameters are explained in Table 2. A rule of thumb: if the inlets of the

parameter	meaning
C	number of components
n	number of inlets
m	number of outlets
ℓ	number of internal variables
d	number of characterizing equations
t	number of thermal equilibrium conditions (if any)

Table 2: Parameters occurring in the degrees of freedom analysis.

unit are known (fixed) then there should be exactly ℓ degrees of freedom left.

6 Model equations of the atomic units

6.1 Divider

Graphical representation: See Figure 4.

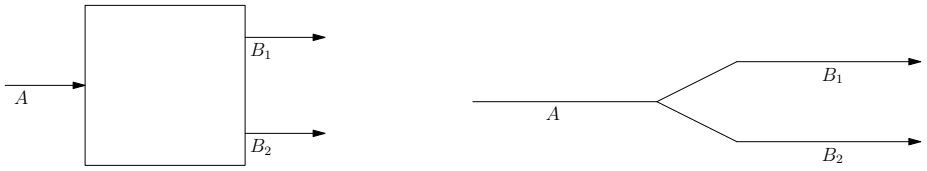

Figure 4: Graphical and simplified graphical representation of the divider (left and right, respectively) with inlet A and outlets B_1 and B_2.

Unit parameter: ζ

Characterizing equations:

$$B_1.f_i = \zeta\, B_2.f_i \qquad (i = 1 : C)$$
$$B_1.H = \zeta\, B_2.H$$

Degrees of freedom:

$$n_{\text{var}} = 3(C+2) + 1$$
$$n_{\text{eq}} = (C+1) + 2 + (C+1)$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = C + 3$$

Degrees of freedom left when all input streams are fixed: 1

Note: The non-negativity bound constraints on the flowrates imply

$$0 \leq \zeta \leq 1.$$

Implementation: See Listing 3.

Listing 3: Implementation of the divider.

```
atomic unit: divider {
  inlets: i
  outlets: o1, o2

  variable: zeta .. real number

  equations {
    o1.f = zeta * o2.f
    o1.H = zeta * o2.H
  }
}
```

6.2 Mixer

Graphical representation: See Figure 5.

Internal variables: None.

Degrees of freedom:

$$n_{\text{var}} = (n+1)(C+2)$$
$$n_{\text{eq}} = (C+1) + 1$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = n(C+2)$$

Degrees of freedom left when all input streams are fixed: 0

Implementation: See Listing 4.

Figure 5: Graphical representation of a mixer with multiple inlets (left), simplified graphical representation of a mixer with two inlets (right).

Listing 4: Implementation of the mixer.

```
atomic unit: mixer {
  parameter: nI .. natural number

  inlets: i[nI]
  outlets: o
}
```

6.2.1 Heat of mixing

Perhaps the simplest realistic case is considered below. A and B are miscible liquid input streams at equal pressures and no phase-transition occurs during mixing; the output stream is C. In this case, the thermodynamically correct heat balance (enthalpy balance) of the mixer is

$$A.H + B.H + H_{\text{mix}}(A, B) = C.H. \tag{3}$$

H_{mix} is called the **heat of mixing**. It is also referred to as the enthalpy of mixing or the excess enthalpy. We have $H_{\text{mix}} = 0$ only for ideal mixtures. Essentially all multicomponent mixtures are non-ideal. H_{mix} can be computed with nonlinear model equations such as the Wilson, NRTL or the UNIQUAC model (Prausnitz et al. [9]), assuming the model parameters have been determined from experimental data. Modeling the mixer becomes significantly more complicated if there are (i) multiple phases in any of the streams A, B and C, (ii) chemical reaction or (iii) phase transition occurs simultaneously with the mixing.

The heat of mixing is almost always neglected without a notice in the chemical engineering literature. The linear heat balance

$$A.H + B.H = C.H$$

is used instead of (3), even if there are multiple phases in any of the streams A, B and C. This obviously simplifies the computations. Besides, a sufficiently accurate model for H_{mix} may not be available due to the lack of experimental data.

The mixer is the only atomic unit having multiple inlets. Thus, a nonlinear mixer has a domino effect: many of the composite units would be no longer worth decomposing.

The error caused by neglecting the heat of mixing may be imperceptible, or it may be considerable (Smith [10], pp. 429–438). Nevertheless, it is customery to use the thermodynamically incorrect linear heat balance even if detailed, thermodynamically consistent models are used in all computations.

6.3 Heat exchanger

Graphical representation: See Figure 2.

Unit parameter: Q

Degrees of freedom:

$$n_{\text{var}} = 2(C+2)+1$$
$$n_{\text{eq}} = (C+1)+1$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = C+3$$

Degrees of freedom left when all input streams are fixed: 1

Implementation: See Listing 5.

Listing 5: Implementation of the heat exchanger.

```
atomic unit: heat exchanger {
  inlets: i
  outlets: o

  drop equations: exchanged heat
}
```

6.4 Pressure changer

Graphical representation: See Figure 2.

Unit parameter: Δp

Degrees of freedom:

$$n_{\text{var}} = 2(C+2)+1+1$$
$$n_{\text{eq}} = (C+1)+1+1$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = C+3$$

Degrees of freedom left when all input streams are fixed: 1

Implementation: See Listing 6.

Listing 6: Implementation of the pressure changer.

```
atomic unit: pressure changer {
  inlets: i
  outlets: o

  drop equations: pressure change
}
```

6.5 Reactor

Graphical representation: See Figure 2.

Unit parameters: $\xi_r \geq 0$ $(r = 1 : R)$.

Degrees of freedom:

$$
\begin{aligned}
n_{\text{var}} &= 2(C+2) + R \\
n_{\text{eq}} &= (C+1) + 1 \\
n_{\text{dof}} &= n_{\text{var}} - n_{\text{eq}} = C + 2 + R
\end{aligned}
$$

Degrees of freedom left when all input streams are fixed: R.

Note: The rate equations

$$RE_r(\xi_r, B.f, p, T) = 0 \quad (r = 1 : R),$$

supplied by the modeler, are typically used if the unit needs to be properly defined.

Implementation: See Listing 7.

Listing 7: Implementation of the reactor.

```
atomic unit: reactor {
  parameter: rateEquations .. subtype of model

  drop equations: rate equations
}
```

6.6 Flash

At present, the model below assumes two phases; see Section 8 how to check this assumption numerically.

Graphical representation: See Figure 2.

Internal variable: T

Characterizing equations:

$$EOS_i(B_1.f, B_2.f, A.p, T) = 0 \quad (i = 1 : C) \;\; \text{(phase equilibrium)}$$
$$EOS(B_1, T) = 0 \;\; \text{(thermal equilibrium)}$$
$$EOS(B_2, T) = 0$$

An important degenerate case is when the equations of state are independent of T and T is not an internal variable:

$$EOS_i(B_1.f, B_2.f, A.p) = 0 \quad (i = 1 : C) \;\; \text{(phase equilibrium)}$$
$$EOS(B_1, B_2) = 0 \;\; \text{(thermal equilibrium)}$$

Degrees of freedom:

$$n_{\text{var}} = 3(C + 2) + 1$$
$$n_{\text{eq}} = (C + 1) + 2 + C + 2$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = C + 2$$

Degrees of freedom left when all input streams are fixed: 0

Notes: If there is, by design, vapor-liquid equilibrium (**VLE**) inside the flash unit then it is called a **VLE flash**. If none of the outlets of a VLE flash have zero flowrate then the liquid phase is at **bubble point** and the vapor phase is at **dew point**, see Section 8.
Traditionally, the inlet is called the Feed and $F = \sum A.f_i$; B_1 is the outlet with the Vapor phase and $V = \sum B_1.f_i$; similarly, B_2 is the outlet with Liquid phase and $L = \sum B_2.f_i$. EOS is expressed with **mole fractions**, traditionally y belongs to the vapor phase and x to the liquid phase, and $y_i = B_1.f_i/V$ and $x_i = B_2.f_i/L$. Usually, the same EOS can be used to determine the **molar enthalpy** h_V and h_L of the vapor and liquid outlets, respectively.

Implementation: See Listing 8 and Section 8.

6.7 Reactive VLE flash

Graphical representation: Same as the flash unit, see Figure 2.

Internal variables: T and unit parameters $\xi_r \geq 0$ $(r = 1 : R)$.

Characterizing equations: Those inherited from the flash unit,

$$EOS_i(B_1.f, B_2.f, A.p, T) = 0 \quad (i = 1 : C) \;\; \text{(phase equilibrium)}$$
$$EOS(B_1, T) = 0 \;\; \text{(thermal equilibrium)}$$
$$EOS(B_2, T) = 0.$$

Listing 8: Implementation of the flash atomic unit.

```
parameters {
  VLE .. subtype of model
  enthalpy .. subtype of model
}

quantities {
  molar enthalpy (J/mol)
}

atomic unit: flash {
  parameters {
    VLEModel .. subtype of model (default: VLE)
    enthalpyModel .. subtype of model (default: enthalpy)
  }

  inlets: i
  outlets: oV, oL

  variables {
    V, L .. molar flowrate
    x[C], y[C] .. real number
    hV, hL .. molar enthalpy
  }

  equations {
    sum(x) = 1
    sum(y) = 1
    oV.f = V*y
    oL.f = L*x
    oV.H = V*hV
    oL.H = L*hL
    VLEModel.equations
    enthalpyModel.equations
  }
}
```

Degrees of freedom:

$$n_{\text{var}} = 3(C+2) + 1 + R$$
$$n_{\text{eq}} = (C+1) + 2 + C + 2$$
$$n_{\text{dof}} = n_{\text{var}} - n_{\text{eq}} = C + 2 + R$$

Degrees of freedom left when all input streams are fixed: R

Note: Typically rate equations

$$RE_r(\xi_r, B_2) = 0 \quad (r = 1 : R)$$

are used if the unit needs to be properly defined. The assumption behind the above rate equations is that the reaction takes place only in the liquid phase and the equations do not depend on B_1 explicitly.

Implementation: See Listing 9.

Listing 9: Implementation of the reactive VLE flash unit.

```
atomic unit: reactive flash extends: flash {
  drop equations: reaction rate
}
```

6.8 Partial reboiler

Same as the the flash or the reactive flash unit but the default $Q = 0$ equation is dropped.

Graphical representation: Same as the flash unit, see Figure 2.

Unit parameters: Q and $\xi_r \geq 0$ $(r = 1 : R)$ in the reactive case.

Characterizing equations: Same as the flash unit except that the default $Q = 0$ equation is dropped.

Degrees of freedom: Same arguments hold as with the flash unit or the reactive VLE flash unit but there is one additional degrees of freedom due to the dropped $Q = 0$ equation.

Degrees of freedom left when all input streams are fixed: 1 in the non-reactive case; $R+1$ in the reactive case., (if the rate equations are supplied in the reactive case).

Implementation: See Listing 10.

Listing 10: Implementation of the partial reboiler.

```
atomic unit: reboiler extends: flash {
  drop equations: exchanged heat
}
```

7 Composite units

Composite units are obtained by connecting atomic units and requiring that certain specifications, if any, are met. The corresponding type of equations are as follows.

Connections with other units: These equations describe how the units are connected by equating the corresponding variables of the involved streams.

Specifications: These equations, if any, make the steady state model properly defined. They usually correspond to closed loop control systems and can show large variation.

Note that no additional equations are needed: The conservation laws are already ensured by the atomic units, adding the balance equations would be redundant (linearly dependent).

7.1 Interpretation of the connecting equations in the implementation

The interpretation of the connecting equations are illustrated on the total condenser, see Figure 6. Additional details of this unit are given in Subsection 7.4 but all the necessary information to understand the connecting equations is given by the figure.

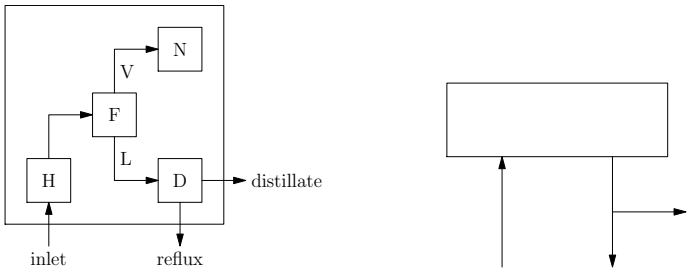

Figure 6: Left: graphical representation of the total condenser; H: heat exchanger; F: flash; V: vapor outlet of the flash; L: liquid outlet of the flash; N: null sink; D: divider. Right: Right: simplified representation; note that the divider is still explicitly shown.

The inlet of the total condenser is the inlet of the heat exchanger; the outlets of the divider are the outlets of the total condenser. This is expressed with the = sign in the connecting equations, see Listing 11 at `connections`.

The other type of the connecting equations describes how the outlets of the internal units are connected to the inlets of another internal unit. This is expressed with the → sign, pointing from the outlet of one internal unit to the inlet of another internal unit; see Listing 11.

Both types of the connecting equations establish that the two streams involved in the equation are equal. If two streams are equal then all their corresponding stream variables are equal.

Listing 11: Implementation of the total condenser.

```
composite unit: total condenser {
  parameter: FlashUnit .. subtype of flash (default: flash)

  inlets: i
  outlets: reflux, distillate

  subunits {
    flash .. variable type FlashUnit
    heatExchanger .. heat exchanger
    divider .. divider
  }

  connections {
    inlet i = heatExchanger.i
    heatExchanger.o -> flash.i
    flash.oL -> divider.i
    flash.oV -> null
    outlet distillate = divider.o1
    outlet reflux = divider.o2
  }
}
```

7.2 VLE stage

Graphical representation: See Figure 7.

Subunits: A mixer and a VLE flash unit (VLE stage) or a reactive VLE flash unit (reactive VLE stage).

Connecting equations: See Listing 12.

Degrees of freedom left when all input streams are fixed: 0.

Note: The outlets are in liquid phase is at bubble point and the vapor phase at dew point, respectively. The flash unit is assumed to operate in the two-phase region, see Section 8.
The inlet V', see Figure 7, is usually the vapor outlet V of another stage or a reboiler; similarly, the inlet L' is usually the liquid outlet L of another

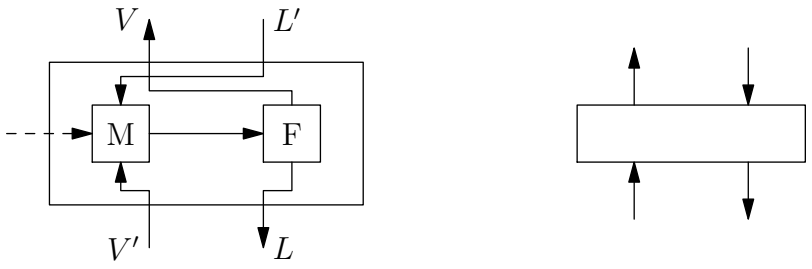

Figure 7: Left: graphical representation of the VLE stage; dashed arrow: optional feed; M: Mixer; F: flash; V: vapor outlet at dew point; L: liquid outlet at bubble point; inlets: V' and L'. Right: simplified representation; the optional feed is not shown.

stage or a condenser. However, it is not assumed that the inlets V' and L' are at dew and bubble point, respectively.

Implementation: See Listing 12.

Listing 12: Implementation of the VLE stage.

```
composite unit: VLE stage {
  % "subtype of" is a metatype, this is a type parameter
  parameter: FlashUnit .. subtype of flash (default: flash)

  inlets: iL, iF (optional), iV
  outlets: oV, oL

  subunits {
    flash .. variable type FlashUnit
    mixer .. mixer (nI = 3)
  }

  connections {
    inlet inlets = mixer.inlets
    mixer.o -> flash.i
    outlet oV = flash.oV
    outlet oL = flash.oL
  }
}
```

7.3 VLE cascade

Figure 8 shows an example of hierarchical decomposition. The vapor-liquid equilibrium cascade is a cascade of stages. A stage is a mixer and a flash unit connected appropriately; see Figure 7. In real life, the stages are the smallest, still functioning pieces. The decomposition of the stage into a mixer and a flash unit is an abstraction, as the stage does not have a mixer or a flash unit inside. Nevertheless, this decomposition is valid for modeling.

Graphical representation: See Figure 8.

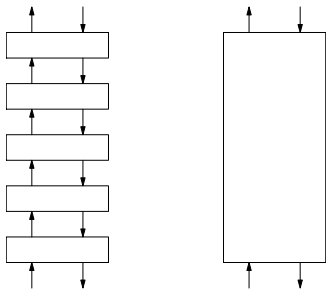

Figure 8: Left: graphical representation of the VLE cascade, optional feeds are not shown. Right: simplified representation; the optional feed is not shown.

Subunits: N pieces of VLE stages.

Connecting equations: See Listing 13.

Degrees of freedom left when all input streams are fixed: 0.

Note: Given that the outlets of the cascade are the outlets of the corresponding stages, the liquid phase outlet is at bubble point and the vapor phase outlet is at dew point.

Implementation: See Listing 13.

7.4 Total condenser

Graphical representation: See Figure 6.

Subunits: A heat exchanger, a flash and a divider.

Specification: 1 equation expressing that the liquid outlet of the flash unit is at bubble point, see Section 8.

Connecting equations: See Listing 11.

Degrees of freedom left when all input streams are fixed: 1.

Note: The **reflux ratio** $R = \zeta$ is often considered as a unit parameter, where ζ is the unit parameter of the divider.

Implementation: See Listing 11.

Listing 13: Implementation of the VLE cascade.

```
composite unit: multiple feed VLE cascade {
  parameters {
    Flash .. subtype of flash (default: flash)
    nStages .. natural number
  }

  inlets: iL, iF[nStages], iV
  outlets: oV, oL

  subunits {
    stages[nStages] .. VLE stage (FlashUnit = Flash)
    % or equivalently: .. VLE stage (FlashUnit := variable type Flash)
    % := takes a type name, = takes a formula
  }

  connections {
    inlet iL = stages[1].iL
    outlet oV = stages[1].oV
    stages[2].oV -> stages[1].iV

    for i in 2:nStages-1 {
      stages[i-1].oL -> stages[i].iL
      stages[i+1].oV -> stages[i].iV
    }

    for i in 1:nStages {
      inlet iF[i] = stages[i].iF
    }

    stages[nStages-1].oL -> stages[nStages].iL
    inlet iV = stages[nStages].iV
    outlet oL = stages[nStages].oL
  }
}
```

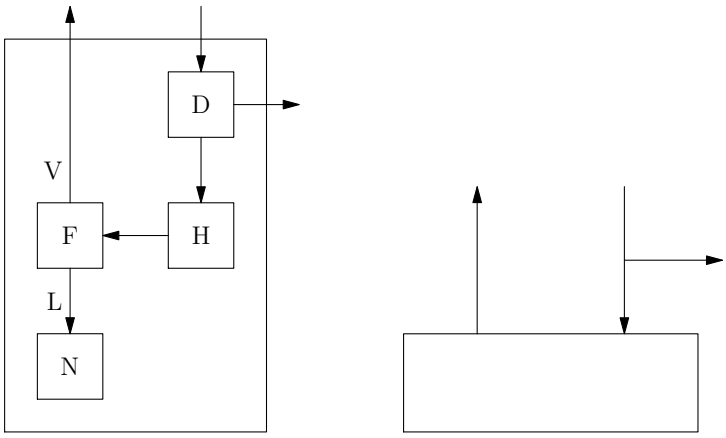

Figure 9: Left: graphical representation of the total reboiler; D: divider; H: heat exchanger; F: flash; L: liquid outlet of the flash; N: null sink; V: vapor outlet of the flash. Right: simplified representation; note that the divider is still explicitly shown.

7.5 Total reboiler

Graphical representation: See Figure 9.

Subunits: A divider, a heat exchanger and a flash.

Specification: 1 equation expressing that the vapor outlet of the flash unit is at dew point, see Section 8.

Connecting equations: See Listing 14.

Degrees of freedom left when all input streams are fixed: 1.

Implementation: See Listing 14.

7.6 A practical example

The decomposition of the units in this work is unconventional in the sense that the units here are different from those found in the chemical engineering literature. For example the equipment in Yi & Luyben [11] referred to as reactor cannot be decomposed further into smaller, functioning pieces. However, it can be modeled by connecting 7 atomic units and a null sink appropriately; see Figure 10. None of these units is the reactor presented in Subsection 6.5.

Listing 14: Implementation of the total reboiler.

```
composite unit: total reboiler {

    subunits {
      divider .. divider
      heatExchanger .. heat exchanger
      flash .. flash
    }

    inlets: iL
    outlets: oV, oBulk

    connections {
      inlet  iL    = divider.i
      outlet oV    = flash.oV
      outlet oBulk = divider.o2
      divider.o1   -> heatExchanger.i
      heatExchanger.o -> flash.i
      flash.oL -> null
    }
}
```

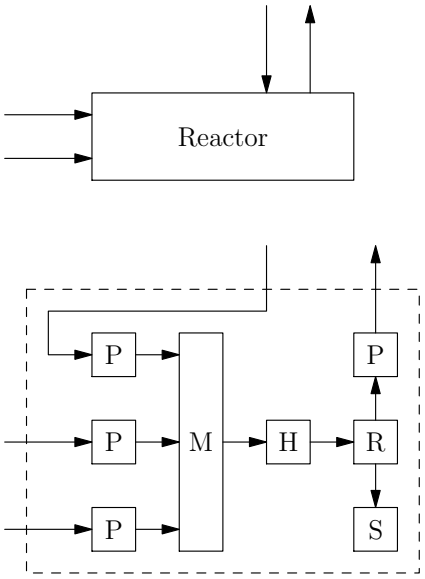

Figure 10: The reactor of Yi & Luyben and its abstract decomposition into atomic units. P: pressure changer, M: mixer, H: heat exchanger, R: reactive flash, S: sink.

8 Flash calculations

Variables and their bound constraints:

$$K_i \geq 0 \quad (i = 1 : C)$$
$$T \geq 0$$
$$0 \leq \lambda \leq 1$$
$$p \geq 0$$
$$0 \leq z_i \leq 1 \quad (i = 1 : C)$$
$$H_F$$

Defined variables:

$$x_i := \tfrac{z_i}{\lambda+(1-\lambda)K_i}$$
$$y_i := K_i x_i$$
$$\psi := \textstyle\sum \tfrac{(K_i-1)z_i}{\lambda+(1-\lambda)K_i} \quad (= \sum y_i - \sum x_i)$$
$$h := h(x, p, T)$$
$$H := H(y, p, T)$$
$$\zeta := \lambda h + (1-\lambda)H - H_F$$

Here, h / H are equations of state determining the liquid / vapor molar enthalpy at bubble / dew point, respectively.

Equations:

$$\textstyle\sum z_i = 1$$
$$\psi = 0$$
$$\zeta = 0$$
$$K_i = K_i(x, y, p, T) \quad (i = 1 : C)$$

Degrees of freedom: $C + 1$. The flash unit has $C + 2$ degrees of freedom but the flash calculations but F (see Section 6.6) only acts like a scaler, hence the $C + 1$ degrees of freedom.

Relating to the flash unit: The following equations relate these variables to the stream variables of the flash unit; see also Section 6.6.

$$A.f_i = Fz_i$$
$$A.p = p$$
$$A.H = FH_F$$
$$B_1.f_i = Vy_i$$
$$B_1.p = p$$
$$B_1.H = VH$$
$$B_2.f_i = Lx_i$$
$$B_2.p = p$$
$$B_2.H = Lh$$

8.1 Notes

The function

$$\psi(\lambda) := \sum \frac{(K_i - 1)z_i}{\lambda + (1 - \lambda)K_i} \tag{4}$$

is monotone in λ,

$$\frac{\mathrm{d}\psi(\lambda)}{\mathrm{d}\lambda} = \sum \frac{(K_i - 1)^2 z_i}{(\lambda + (1 - \lambda)K_i)^2}.$$

The $\psi(0)$ and $\psi(1)$ values tell the state of the phase(s); see Table 3.

$\psi(0)$	$\psi(1)$	$\psi(0)\psi(1)$	state
< 0	< 0	> 0	sub-cooled liquid
< 0	$= 0$	$= 0$	liquid at bubble point
< 0	> 0	< 0	vapor and liquid phases
$= 0$	> 0	$= 0$	vapor at dew point
> 0	> 0	> 0	superheated vapor

Table 3: State of the phase(s) in the flash calculations given by $\psi(\lambda)$ in Equation (4).

8.2 Degenerate cases

For ideal mixtures $K_i = K_i(p, T)$ holds. If only the vapor phase can be approximated as ideal (e.g. vapor at atmospheric pressure usually can be) then $K_i = K_i(x, p, T)$. Since distillation columns are often operated at atmospheric pressure, the latter case has practical significance.

In case of ideal binary mixtures, only one component is used to describe the composition of the mixture as $x_2 = 1 - x_1$ and $y_2 = 1 - y_1$; and we have $K_1 = K_1(x_1, p)$. If the heat balance is neglected, that is, $\zeta = 0$ is dropped, then T is not considered as a variable.

Another practically relevant case is when the temperature dependence of h and H is neglected: $h := h(x, p)$ and $H := H(y, p)$.

8.3 A practical example of the phase equilibrium condition

According to the modified Raoult-Dalton equation

$$y_i p = \gamma_i x_i p_i^{\mathrm{sat}}$$

where p_i^{sat} can be computed, for example, with the Antoine equations

$$\log_{10} p_i^{\text{sat}} = A_i - \frac{B_i}{T + C_i},$$

with substance specific constants A_i, B_i, C_i. This gives

$$K_i(x, p, T) \equiv \frac{y_i}{x_i} = \frac{p}{\gamma_i(x, T)\, p_i^{\text{sat}}(T)},$$

where γ_i is called the **activity coefficient**.

9 Activity coefficient models

Practical examples of Equations (2) used in the phase equilibrium conditions are given in this section.

In all of these models, the activity coefficient γ_i is a nonlinear function of the compositions x and the temperature T but independent of the pressure p,

$$\ln \gamma_i = f_i(x, T) \qquad (i = 1 : C).$$

The indices go from 1 to C in all the equations below, unless otherwise stated.

9.1 Wilson model

Parameters: $V_i,\ \lambda'_{ij} = \lambda_{ij} - \lambda_{ii},\ R$

Equations:

$$\begin{aligned}
&\Lambda_{ij} = \tfrac{V_j}{V_i} \exp\left(-\tfrac{\lambda'_{ij}}{RT}\right) \\
&\ln \gamma_i = -\ln\left(\textstyle\sum_j \Lambda_{ij} x_j\right) + 1 - \sum_k \tfrac{x_k \Lambda_{ki}}{\sum_j \Lambda_{kj} x_j}
\end{aligned}$$

Note: All Λ_{ij} are strictly positive by definition.

9.2 NRTL

Parameters: $g'_{ij} = g_{ij} - g_{jj},\ \alpha_{ij},\ R$

Equations:

$$\begin{aligned}
&\tau_{ij} = \tfrac{g'_{ij}}{RT} \\
&G_{ij} = \exp(-\alpha_{ij} \tau_{ij}) \\
&\ln \gamma_i = \tfrac{\sum_j \tau_{ji} G_{ji} x_j}{\sum_k G_{ki} x_k} + \sum_j \tfrac{x_j G_{ij}}{\sum_k G_{kj} x_k} \left(\tau_{ij} - \tfrac{\sum_\ell x_\ell \tau_{\ell j} G_{\ell j}}{\sum_k G_{kj} x_k}\right)
\end{aligned}$$

9.3 Extended NRTL (a particular one)

Additional parameters: Δg_{ijk}

The changed equation:

$$\tau_{ij} = \frac{g_{ij} - g_{jj} + \sum_k x_k \Delta g_{ijk}}{RT}$$

9.4 UNIQUAC (a particular formulation)

Parameters: $a_{ij},\ r_i,\ q_i,\ q_i',\ z$

Equations:

$$\tau_{ij} = \exp\left(-\tfrac{a_{ij}}{T}\right)$$
$$\Phi_i = \tfrac{r_i x_i}{\sum_j r_j x_j}$$
$$\theta_i = \tfrac{q_i x_i}{\sum_j q_j x_j}$$
$$\theta_i' = \tfrac{q_i' x_i}{\sum_j q_j' x_j}$$
$$l_i = \tfrac{z}{2}(r_i - q_i) - (r_i - 1)$$
$$\ln \gamma_i = \ln \gamma_i^{\text{comb}} + \ln \gamma_i^{\text{res}}$$
$$\ln \gamma_i^{\text{comb}} = \ln \tfrac{\Phi_i}{x_i} + \tfrac{z}{2} q_i \ln \tfrac{\theta_i}{\Phi_i} + l_i - \tfrac{\Phi_i}{x_i} \sum_j x_j l_j$$
$$\ln \gamma_i^{\text{res}} = q_i' \left(1 - \ln \textstyle\sum_j \tau_{ji} \theta_j' - \sum_k \tfrac{\theta_k' \tau_{ik}}{\sum_j \tau_{jk} \theta_j'}\right)$$

Note: An alternative form of $\ln \gamma_i^{\text{comb}}$ is

$$\ln \gamma_i^{\text{comb}} = \ln \frac{\Phi_i}{x_i} - \frac{\Phi_i}{x_i} + 1 - \frac{z}{2} q_i \left(1 + \ln \frac{\Phi_i}{\theta_i} - \frac{\Phi_i}{\theta_i}\right)$$

10 Separation operations

A chemical plant takes raw materials as input and produces products as output. Roughly speaking, three steps can be distinguished in a chemical plant: preparation, reaction and purification. See Figure 11. Unwanted chemical substances are separated from the raw input materials in the first step. The unwanted substances may interfere with the reaction in the second step. The reaction produces the desired products and byproducts. Usually a significant fraction of the reactants remain unreacted. These unreacted reactants, the products and the waste byproducts are separated in the third step, called the purification step. The unreacted reactants are recycled, that is, they are fed back to the first step.

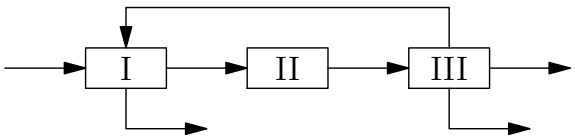

Figure 11: Schematic figure of a chemical plant. Input: raw materials, output: unwanted materials, products and byproducts. The steps are (I) preparation, (II) reaction and (III) purification.

Both the first and the third step involves separation operations. In a typical chemical plant, 40–80% of the investment is spent on separation operation equipments (Prausnitz et al. [9], p. 2).

Many of the practically relevant equipments used in separation operations (multistage extraction, absorption, desorption, stripping and distillation) are internally a cascade. Not surprisingly, their mathematical model can be solved in a sequential manner.

Identifying multiple steady states is critical to proper design, simulation, control, and operation of these equipments. Unfortunately, professional simulators return only one solution at a time, without indicating the possible existence of other solutions. Usually, only one of the steady-states is desired, the so-called high purity branch. The other steady states are undesirable and potentially harmful as they can lead to unexpected behavior, meaning that the equipment may respond to perturbation in a counterintuitive way.

10.1 Internal physical structure of distillation columns

Distillation columns are used in separation operations. The body of a multistage distillation column is a cascades of stages. In the cascade, the output of one stage is the input of its two neighbors and vice versa, see Figure 8. This structural information can be exploited to solve the underlying process model efficiently.

The internal physical structure is reflected in the mathematical model of the columns. The equations can be evaluated in a sequential manner after guessing just a few variables at one end of the cascade. The essential dimension of the problem is given by the number of variables that have to be guessed to start the stage-by-stage computations. The steady-state model of distillation columns are essentially low-dimensional even if their steady-state model is large-scale.

This approach, reducing the large-scale model to a low-dimensional one, is called the stage-by-stage calculation (Lewis & Matheson [7]). Unfortunately, solving the low-dimensional model is very difficult if not impossible with this method, as it shows an extreme sensitivity to the initial estimates. Thus, currently only high-dimensional techniques are in use (Doherty et al. [4], 13–33). But a proof-of-concept method remedies the numerical difficulties of the stage-by-stage calculation, see Baharev & Neumaier [2].

10.2 Example: multiple steady-states in ideal two-product distillation

The implementation is tested on the distillation column presented in Jacobsen & Skogestad [6]. Its main structure corresponds to the linear structure presented in Figure 8, and detailed in subsection 10.1.

Perhaps the simplest distillation columns are the single feed two-product columns with ideal vapor-liquid equilibrium. Even these columns can have multiple stead-states (Jacobsen & Skogestad [6]). One type of multiplicity can occur when the column has its input specified on a mass or volume basis (e.g., mass reflux and molar boilup). This is of high practical relevance as industrial columns usually have their inputs specified in this way.

The model equations are taken from Baharev et al. [1]. Specifications are: methanol-propanol feed composition, mass reflux flow rate and vapor molar flow rate of the boilup. Heat balances are included in the model.

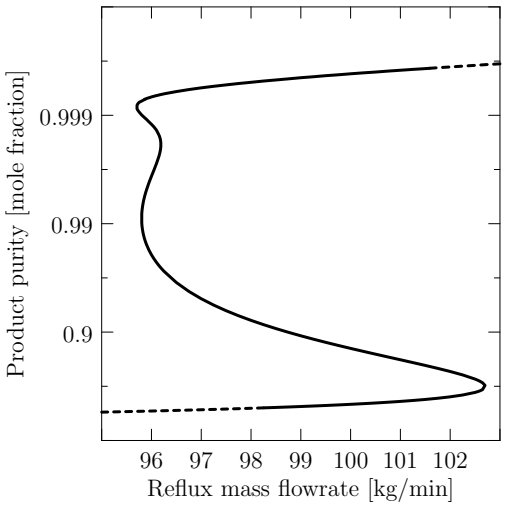

Figure 12: Bifurcation diagram, multiple steady-states in ideal two-product distillation. The infeasible steady-states are represented by dashed lines.

In many studies, one is interested in the dependence of the characteristics on a design parameter (the bifurcation parameter) that can be varied, resulting in bifurcation diagrams. In this case, the design parameter is the reflux flowrate specified on mass basis, and the observed parameter is the product purity. The bifurcation diagram is given in Figure 12. The model equations have five distinct solutions in a certain range of the reflux flow rate. One of the solutions is infeasible in practice because it would result in negative flow rates.

11 Test examples

11.1 Reactive distillation column for manufacturing ethylene glycol

The steady state model of a reactive distillation column for ethylene glycol synthesis is presented here, taken from Ciric & Miao [3]. The second reaction is neglected as in subsection *7.3. Multiple reactions* of that paper. The index for the reactions is omitted since only one reaction remains. The implementation of the model is given in the Appendix.

11.1.1 Model description

Parameters

C	number of components	$C = 3$
N	number of stages	$N = 10$
f_{ij}	feed flow rates	$i = 1 : C, \quad j = 1 : N$
ν_i	stoichiometric coefficients	$i = 1 : C$
λ	homotopy parameter	
W_j	reaction volume	$j = 1 : N$
H_{vap}	heat of vaporization	
H_r	heat of reaction	
β	reboiler boil-up ratio	

Variables

x_{ij}	liquid phase composition	$i = 1 : C, \quad j = 1 : N + 1$
y_{ij}	vapor phase composition	$i = 1 : C, \quad j = 0 : N$
l_{ij}	liquid phase component molar flowrate	$i = 1 : C, \quad j = 1 : N + 1$
v_{ij}	vapor phase component molar flowrate	$i = 1 \ldots C, \quad j = 0 : N$
L_j	liquid phase molar flowrate	$j = 1 : N + 1$
V_j	vapor phase molar flowrate	$j = 0 : N$
T_j	temperature at stage j	$j = 1 : N$
ξ_j	extent of reaction at stage j	$j = 1 : N$

Equations

Molar flowrates to simplify the notation

$$l_{ij} := x_{ij} L_j \quad \text{for } i = 1 : C, \quad j = 1 : N + 1 \tag{5}$$

$$v_{ij} := y_{ij} V_j \quad \text{for } i = 1 : C, \quad j = 0 : N \tag{6}$$

Condenser

$$L_{N+1} = V_N \tag{7}$$

$$x_{i,N+1} = y_{i,N} \quad \text{for } i = 1 : C \tag{8}$$

Reboiler

$$V_0 = \beta L_1 \quad (\beta \text{ from specification}) \tag{9}$$

$$y_{i,0} = x_{i,1} \quad \text{for } i = 1 : C \tag{10}$$

Phase equilibrium

$$y_{ij} = K_i(T_j)x_{ij} \quad \text{for } i = 1 : C, \quad j = 1 : N \tag{11}$$

Extent of reaction

$$\xi_j = \lambda W_j f(x_{1j}, \ldots, x_{C,j}, T_j) \quad \text{for } j = 1 : N \tag{12}$$

Material balances

$$f_{ij} + v_{i,j-1} + l_{i,j+1} + \nu_i \xi_j = v_{ij} + l_{ij} \quad \text{for } i = 1 : C, \quad j = 1 : N \tag{13}$$

Heat balances

$$H_{\text{vap}}(V_{j-1} - V_j) = H_r \xi_j \quad \text{for } j = 1 : N \tag{14}$$

Summation equations

$$\sum_{i=1}^{C} x_{ij} = 1 \quad \text{for } j = 1 : N \tag{15}$$

$$\sum_{i=1}^{C} y_{ij} = 1 \quad \text{for } j = 1 : N \tag{16}$$

11.1.2 Elimination order

Let introduce the following notation

$$\xi_{\text{tot}} := \sum_j \xi_j \quad \text{(overall extent of reaction).} \tag{17}$$

Once a value for ξ_{tot} is assumed, everything else can be computed by solving univariate equations. This makes the problem essentially 1-dimensional. However, solving it as a 1-dimensional zero-finding problem does not work. The equations show extreme sensitivity to the value of ξ_{tot}.

Let

$$b_i := l_{i,1} - v_{i,0} \quad \text{for } i = 1 : C \quad \text{(bulk component flowrate).} \tag{18}$$

The sum of all equations (13) over $j = 1 : N$, also taking into account (5)–(8) and (15), (16), is

$$\sum_j f_{ij} = l_{i,1} - v_{i,0} - \nu_i \sum_j \xi_j \quad \text{for } i = 1 : C. \tag{19}$$

Once a value is assumed for $\xi_{\mathbf{tot}}$ the elimination is started as follows. Solve (19) for the b_i:

$$b_i = \sum_{j=1}^{N} f_{ij} + \nu_i \xi_{\text{tot}} \quad \text{for } i = 1 : C. \tag{20}$$

From (5), (6) and (9), (10) we have

$$l_{i,1} = \frac{1}{1-\beta} b_i \quad \text{for } i = 1 : C,$$

$$v_{i,0} = \frac{\beta}{1-\beta} b_i \quad \text{for } i = 1 : C.$$

At this point, everything is known to start the stage-by-stage propagation, working from $j = 1$ to $j = N$.

Find $l_{i,j+1}$ and $v_{i,j}$ given l_{ij} and $v_{i,j-1}$ (going from stage j to $j + 1$). First,

$$x_{ij} = \frac{l_{ij}}{\sum_i l_{ij}}. \tag{21}$$

Solve

$$\sum_i K_i(T_j) x_{ij} = 1 \tag{22}$$

for T_j. Then perform the elimination in the order given below.

$$y_{ij} = K_i(T_j) x_{ij} \quad \text{for } i = 1 : C \tag{23}$$

$$\xi_j = W_j f(x_{1j}, \ldots, x_{C,j}, T_j) \tag{24}$$

$$V_{j-1} = \sum_i v_{i,j-1} \tag{25}$$

$$V_j = -\frac{H_r \xi_j}{H_{\text{vap}}} + V_{j-1} \tag{26}$$

$$v_{ij} = y_{ij} V_j \quad \text{for } i = 1 : C \tag{27}$$

$$l_{i,j+1} = v_{ij} + l_{ij} - \nu_i \xi_j - (f_{ij} + v_{i,j-1}) \quad \text{for } i = 1 : C \tag{28}$$

Finally,

$$\sum_i l_{i,N+1} = \sum_i v_{i,N}. \tag{29}$$

This is the final equation, depending only on ξ_{tot}.

11.2 Column of Jacobsen and Skogestad

Steady-state model of an ideal two-product distillation column. This benchmark originates from Jacobsen & Skogestad [6]. The problem has 5 solutions. The implementation is available in the Appendix.

12 Appendix

Unit library implementation

```
% UnitLibrary.cpm: Library of ChemProcMod atomic and basic composite
units

parameters {
  C .. natural number   % number of chemical substances
  VLE .. subtype of model
  enthalpy .. subtype of model
}

quantities {
  molar flow rate (mol/s) >= 0
  pressure (Pa) >= 0
  reaction heat rate (J/s)
  enthalpy flow rate (J/s)
  molar enthalpy (J/mol)
  temperature (K)
}

stream {
  f[C] .. molar flow rate
  p .. pressure
  H .. enthalpy flow rate
}

% the unnamed atomic unit carries the base equations
atomic unit {
  variables {
    reactionRate[C] .. molar flowrate
    reactionHeat .. reaction heat rate
    exchangedHeat .. enthalpy flow rate
    pressureDifference .. pressure

    p .. pressure
  }

  % named equation sets can be overridden
  equations: rate equations {
    reactionRate[1:C] = 0
    reactionHeat = 0
  }

  equations: exchanged heat {
    exchangedHeat = 0
  }

  equations: pressure change {
    pressureDifference = 0
  }

  % the unnamed equation set is invariant and can only be added to
  equations {
    for i in 1:C {
       sum(inlets[j].f[i] for j in 1:nInlets) = ...
         sum(outlets[k].f[i] for k in 1:nOutlets) + reactionRate[i]
    }

    sum(inlets[j].H for j in 1:nInlets) = ...
      sum(outlets[k].H for k in 1:nOutlets) + reactionHeat +
exchangedHeat

    p = min(inlets[j].p for j in 1:nInlets) + pressureDifference
    for k in 1:nOutlets {
      outlets[k].p = p
    }
```

```
  }
}

atomic unit: heat exchanger {
  inlets: i
  outlets: o

  drop equations: exchanged heat
}

atomic unit: pressure changer {
  inlets: i
  outlets: o

  drop equations: pressure change
}

atomic unit: divider {
  inlets: i
  outlets: o1, o2

  variable: zeta .. real number

  equations {
    o1.f = zeta * o2.f
    o1.H = zeta * o2.H
  }
}

atomic unit: reactor {
  parameter: rateEquations .. subtype of model

  drop equations: rate equations
}

atomic unit: flash {
  parameters {
    VLEModel .. subtype of model (default: VLE)
    enthalpyModel .. subtype of model (default: enthalpy)
  }

  inlets: i
  outlets: oV, oL

  variables {
    V, L .. molar flowrate
    x[C], y[C] .. real number
    hV, hL .. molar enthalpy
  }

  equations {
    sum(x) = 1
    sum(y) = 1
    oV.f = V*y
    oL.f = L*x
    oV.H = V*hV
    oL.H = L*hL
    VLEModel.equations
    enthalpyModel.equations
  }
}

atomic unit: reactive flash extends: flash {
  drop equations: rate equations
}

atomic unit: mixer {
  parameter: nI .. natural number
```

```
  inlets: i[nI]
  outlets: o
}

% Composite units
%=====================================================================

composite unit: VLE stage {
  % "subtype of" is a metatype, this is a type parameter
  parameter: FlashUnit .. subtype of flash (default: flash)

  inlets: iL, iF (optional), iV
  outlets: oV, oL

  subunits {
    flash .. variable type FlashUnit
    mixer .. mixer (nI = 3)
  }

  enum: L, F, V

  connections {
    inlet inlets = mixer.inlets
    % same as:
    %inlet iL = mixer.i[L]
    %inlet iF = mixer.i[F]
    %inlet iV = mixer.i[V]
    mixer.o -> flash.i
    outlet oV = flash.oV
    outlet oL = flash.oL
  }
}

composite unit: single feed VLE cascade {
  parameters {
    Flash .. subtype of flash (default: flash)
    nStages .. natural number
    feedStage .. natural number
  }

  inlets: iL, iF, iV
  outlets: oV, oL

  subunits {
    stages[nStages] .. VLE stage (FlashUnit = Flash)
    % or equivalently: .. VLE stage (FlashUnit := variable type Flash)
    % := takes a type name, = takes a formula
  }

  connections {
    inlet iL = stages[1].iL
    outlet oV = stages[1].oV
    stages[2].oV -> stages[1].iV

    for i in 2:nStages-1 {
      stages[i-1].oL -> stages[i].iL
      stages[i+1].oV -> stages[i].iV
    }

    inlet iF = stages[feedStage].iF

    stages[nStages-1].oL -> stages[nStages].iL
    inlet iV = stages[nStages].iV
    outlet oL = stages[nStages].oL
  }
}
```

```
composite unit: multiple feed VLE cascade {
  parameters {
    Flash .. subtype of flash (default: flash)
    nStages .. natural number
  }

  inlets: iL, iF[nStages], iV
  outlets: oV, oL

  subunits {
    stages[nStages] .. VLE stage (FlashUnit = Flash)
    % or equivalently: .. VLE stage (FlashUnit := variable type Flash)
    % := takes a type name, = takes a formula
  }

  connections {
    inlet iL = stages[1].iL
    outlet oV = stages[1].oV
    stages[2].oV -> stages[1].iV

    for i in 2:nStages-1 {
      stages[i-1].oL -> stages[i].iL
      stages[i+1].oV -> stages[i].iV
    }

    for i in 1:nStages {
      inlet iF[i] = stages[i].iF
    }

    stages[nStages-1].oL -> stages[nStages].iL
    inlet iV = stages[nStages].iV
    outlet oL = stages[nStages].oL
  }
}

composite unit: total condenser {
  parameter: FlashUnit .. subtype of flash (default: flash)

  inlets: i
  outlets: reflux, distillate

  subunits {
    flash .. variable type FlashUnit
    heatExchanger .. heat exchanger
    divider .. divider
  }

  connections {
    inlet i = heatExchanger.i
    heatExchanger.o -> flash.i
    flash.oL -> divider.i
    flash.oV -> null
    outlet distillate = divider.o1
    outlet reflux = divider.o2
  }
}

composite unit: total reboiler {

    subunits {
      divider .. divider
      heatExchanger .. heat exchanger
      flash .. flash
    }

    inlets: iL
    outlets: oV, oBulk
```

```
    connections {
      inlet  iL    = divider.i
      outlet oV    = flash.oV
      outlet oBulk = divider.o2
      divider.o1   -> heatExchanger.i
      heatExchanger.o -> flash.i
      flash.oL -> null
    }
}
```

Implementation of the reactive distillation column

```
fixed parameters {
  C .. natural number = 3
  nu[C] .. integer = { -1, -1, 1 }
}

model: custom VLE {

  fixed parameters {
    A[1:C,1:4] .. real number = ...
        {{71.9,  5.72,  469,  35.9},...
         {221.2, 6.31,  647,  52.9},...
         {77,    9.94,  645,  71.4}}
  }

  equations {

    for i in 1:C {
      K[i] = A[i,1]*exp(A[i,2]*((T-A[i,3])/(T-A[i,4])))
      y[i] = K[i]*x[i]
    }
  }

}

model: enthalpy {

  fixed parameter: Hvap .. real number = 40.6

  equations {
    hV = Hvap
    hL = 0.0
  }

}

import: UnitLibrary (C = C, VLE := custom VLE, enthalpy := enthalpy)

%=============================================================================

fixed parameters {
  nStages .. natural number = 10
  p .. real number = 1 % dummy pressure
  hF .. real number = 0 % enthalpy of feed
  lambda .. real number = 10
  f[1:C,1:nStages] .. real number = ...
    {default = 0, ...
     [1,5:10] = {4.89, 4.76, 4.69, 4.97, 0.2, 8.05}, ...
     [2,10] = 27.56}

  W[1:nStages] .. real number = ...
    {default = 0, [5:10] = {0.551, 0.481, 0.447, 0.371, 1.447, 0.011}}

}

atomic unit: custom reactive flash extends: reactive flash {

  parameters {
    W .. real number (default value: 0.0)
  }

  variables {
    T    .. temperature >= 270, <=490
    K[C] .. real number >= 0.0
    eta  .. real number >= 0.0
  }
```

```
  fixed parameter: Hr .. real number = -80.0

  equations {

    eta = W*exp(37.0-(9547.7/T))*x[1]*x[2]

    for i in 1:C {
      reactionRate[i] = - nu[i]*eta
    }

    reactionHeat = Hr*eta
  }
}

process: reactive distillation {

  import fixed parameters: RD_params

  subunits {
    condenser .. total condenser
    cascade .. multiple feed VLE cascade (Flash := custom reactive
flash, nStages = 10)
    reboiler .. total reboiler
  }

  sources: feeds[nStages]
  sinks: bulkSink

  parameter specifications {
    cascade.stages[:].FlashUnit.W = lambda*W[:]
  }

  specifications {
    for i in 1:nStages {
      feeds[i].{
        f = f[:,i]
        p = p
        hF = hF
      }
      % or: feeds[i].{f = f[:,i], p = p, hF = hF}
    }
  }

  specifications {
    condenser.divider.zeta = 0.0
    reboiler.divider.zeta = 0.958/(1-0.958)
  }

  connections {
    cascade.oV -> condenser.i
    condenser.reflux -> cascade.iL

    for i in 1:nStages {
      feeds[i] -> cascade.iF[i]
    }

    reboiler.oV -> cascade.iV
    cascade.oL -> reboiler.i
    reboiler.oBulk -> bulkSink
  }
}
```

Implementation of the column of Jacobsen and Skogestad

```
fixed parameters {
  C .. natural number = 2
  MolWeight[C] .. real number = {32.04, 60.10}
}

model: ideal VLE {
  fixed parameter: alpha .. real number = 3.55

  equations {
    y[1] = alpha*x[1]/(1.0+(alpha-1.0)*x[1])
  }
}

model: enthalpy {
  equations {
    hV = 0.1349*exp(-3.98*x[1])+0.4397*exp(-0.088*x[1])
    hL = 0.1667*exp(-1.087*x[1])
  }
}

import: UnitLibrary (C = C, VLE := ideal VLE, enthalpy := enthalpy)

atomic unit: reboiler extends: flash {
  drop equations: exchanged heat
}

%=================================================================

process: Jacobsen test {
  sources: feed(f = {0.5, 0.5}, p = 1.0, H = 0.1667*exp(-1.087*0.5))
  subunits {
    condenser .. total condenser
    cascade .. single feed VLE cascade (nStages = 9, feedStage = 5)
    reboiler .. reboiler
  }
  sinks: distillateSink, liquidSink

  specifications {
    sum(condenser.reflux.f[i]*MolWeight[i] for i in 1:C) = 96.0
    reboiler.V = 3.0
  }

  connections {
    cascade.oV -> condenser.i
    condenser.distillate -> distillateSink
    condenser.reflux -> cascade.iL

    feed -> cascade.iF

    reboiler.oV -> cascade.iV
    cascade.oL -> reboiler.i
    reboiler.oL -> liquidSink
  }
}
```

Concise type sheet

```
! Chemical Process Modeling
! -------------------------
!
! Kevin Kofler
!
! April 30, 2013
!
! This is a grammar for a chemical process modeling language designed
! together
! with Ali Baharev and Arnold Neumaier.



! - the newline character, escaped as &n,
!
! this leads to an unsupported nested context-sensitive constraint.


! A line is a string without newline characters
!
union> String


! Comments start with a % sign and optional blanks and end with a
! newline.
! Stored is only the string in between, without the leading blanks.
!
allOf> text=Line


! Blank line
optional> comment=Comment


! Identifier, can contain only letters, digits and _ and not start with
! a digit
union> String


allOf> name=Identifier
optional> qual=Reference
optional> arrayIndex=ArrayIndex


! Positive integer (see also unary minus)
union> Integer


! Positive floating-point number (see also unary minus)
union> Double


! Keyword "default"
nothing>


! Array index
union> ExpressionLink


! Position indication for an array element
union> ArrayIndex, Default


! Initializer for one or more array elements
allOf> expression=Expression
optional> position=ArrayElementPosition
```

```
! Linked list of array elements
allOf> element=ArrayElement
optional> next=ArrayElementLink


! Array
optional> elements=ArrayElementLink


! Bracketed expression
allOf> expression=Expression


! Function argument
union> Expression


! List of function arguments
allOf> argument=FunctionArgument
optional> next=FunctionArgumentLink


! Function call
allOf> function=Identifier
optional> arguments=FunctionArgumentLink


! Operation over a range
allOf> op=Identifier
allOf> expression=Expression
allOf> counter=Identifier
allOf> range=Range


! Atomic expression
union> BracketedExpression, Reference, FunctionCall, RangeOp, PosInt,
PosFloat, Array


! Binary *
allOf> product=ProductExpression
allOf> factor=AtomicExpression


! Binary /
allOf> product=ProductExpression
allOf> factor=AtomicExpression


! Product expression
union> TimesExpression, DivExpression, AtomicExpression


! Binary +
allOf> sum=SumExpression
allOf> term=ProductExpression


! Binary -
allOf> sum=SumExpression
allOf> term=ProductExpression


! Unary -
allOf> term=ProductExpression
```

```
! Sum expression
union> PlusExpression, MinusExpression, UnaryMinus, ProductExpression


! Range
optional> min=SumExpression
optional> max=SumExpression


! Expression
union> SumExpression, Range


! Linked list of expressions
allOf> expression=Expression
optional> next=ExpressionLink


! Type name, can contain words and non-trailing spaces
! The following types shall be considered builtin types during semantic
! subtype of TYPENAME - a type parameter which can contain any subtype
! of
!                       TYPENAME (the type itself, not a value of that
! type)
! variable type IDENTIFIER - a value of the type contained in the type
! parameter
!                             IDENTIFIER
! natural number - an integer >= 0 (of a given finite precision)
! integer - any integer (of a given finite precision)
! real number - any floating-point number (of a given finite precision)
union> String


! Value specification
union> TypeSpec, Expression


! Parameter specification
allOf> name=Identifier
allOf> value=ValueSpec


! List of parameter specifications
allOf> parameter=ParameterSpec
optional> next=ParameterSpecLink


! Type specification
allOf> name=TypeName
optional> parameters=ParameterSpecLink


! Default Value specification
union> TypeSpec, Expression


! Parameter definition
allOf> name=Identifier
allOf> type=TypeSpec
optional> arrayIndex=ArrayIndex
optional> default=DefaultValueSpec
optional> comment=Comment


union> ParameterDef
```

```
! parameters { block element
union> BlankLine, ParameterDef


! List of parameters { block elements
allOf> element=ParametersElement
optional> next=ParametersElementLink


! parameters { block
optional> startComment=Comment
optional> parameters=ParametersElementLink
optional> endComment=Comment


! Fixed parameter definition
allOf> name=Identifier
allOf> type=TypeSpec
allOf> value=ValueSpec
optional> arrayIndex=ArrayIndex
optional> comment=Comment


union> FixedParameterDef


! fixed parameters { block element
union> BlankLine, FixedParameterDef


! List of fixed parameters { block elements
allOf> element=FixedParametersElement
optional> next=FixedParametersElementLink


! fixed parameters { block
optional> startComment=Comment
optional> fixedParameters=FixedParametersElementLink
optional> endComment=Comment


! Import a file which can only contain fixed parameters, no other
! contents.
! Unlike a general import, this can also appear within units (except
flexible
! units) or processes.
allOf> import=TypeSpec
optional> comment=Comment


! Quantity definition
allOf> name=TypeName
allOf> unit=Expression
optional> minValue=Expression
optional> maxValue=Expression
optional> comment=Comment


union> QuantityDef


! quantities { block element
union> BlankLine, QuantityDef


! List of quantities { block elements
allOf> element=QuantitiesElement
optional> next=QuantitiesElementLink
```

```
! quantities { block
optional> startComment=Comment
optional> quantities=QuantitiesElementLink
optional> endComment=Comment


! Variable
allOf> name=Identifier
optional> arrayIndex=ArrayIndex


! List of variables
allOf> variable=Variable
optional> next=VariableLink


! Variable definition
allOf> variables=VariableLink
allOf> type=TypeSpec
optional> minValue=Expression
optional> maxValue=Expression
optional> comment=Comment


union> VariableDef


! variables { block element
union> BlankLine, VariableDef


! List of variables { block elements
allOf> element=VariablesElement
optional> next=VariablesElementLink


! variables { block
optional> startComment=Comment
optional> variables=VariablesElementLink
optional> endComment=Comment


! Enumeration element
union> Identifier


! List of enumeration elements
allOf> element=EnumElement
optional> next=EnumElementLink


allOf> elements=EnumElementLink
optional> comment=Comment


! General element
! Can appear either at the top level or in models, units (except
! flexible units)
! and processes
union> BlankLine, ParameterLine, ParametersBlock, FixedParameterLine,
FixedParametersBlock, ImportFixedParametersLine, QuantityLine,
QuantitiesBlock, VariableLine, VariablesBlock, EnumLine


! Allowed only at the top level.
allOf> import=TypeSpec
```

```
optional> comment=Comment


! stream { block element
union> BlankLine, VariableDef


! List of stream { block elements
allOf> element=StreamElement
optional> next=StreamElementLink


! stream { block
optional> name=TypeName
optional> startComment=Comment
optional> variables=StreamElementLink
optional> endComment=Comment


! = relation sign
nothing>


! != relation sign
nothing>


! > relation sign
nothing>


! >= relation sign
nothing>


! < relation sign
nothing>


! <= relation sign
nothing>


! Relation sign
union> EqRelation, NeqRelation, GtRelation, GeqRelation, LtRelation,
LeqRelation


! Relation
allOf> lhs=Expression
allOf> sign=RelationSign
allOf> rhs=Expression


! List of relations
allOf> relation=Relation
optional> next=RelationLink


! Relation line
allOf> relation=Relation
optional> comment=Comment


! List of relation lines
allOf> line=RelationLine
optional> next=RelationLineLink
```

```
! With line
! Specifies a list of relations with a common reference at the left hand
! side.
! This is the one-line version with comma-separated relations.
allOf> reference=Reference
optional> relations=RelationLink
optional> comment=Comment


! With block
! Specifies a list of relations with a common reference at the left hand
! side.
! This is the block version with each relation in a separate line.
allOf> reference=Reference
optional> startComment=Comment
optional> relations=RelationLineLink
optional> endComment=Comment


! Reference line
allOf> reference=Reference
optional> comment=Comment


! equations { block element
union> BlankLine, RelationLine, WithLine, WithBlock, ReferenceLine,
IfBlock, ForBlock


! List of equations { block elements
allOf> element=EquationsElement
optional> next=EquationsElementLink


! equations { block
optional> name=TypeName
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> endComment=Comment


! else { block
optional> comment=Comment
optional> equations=EquationsElementLink


! Bracketed relational boolean expression
allOf> rbe=RBE


! Atomic relational boolean expression
union> Relation, BracketedRBE


! Boolean not
allOf> atom=AtomicRBE


! Elementary relational boolean expression
union> AtomicRBE, BooleanNot


! Boolean and
allOf> conjunction=ConjunctionalRBE
allOf> term=ElementaryRBE
```

```
! Conjunctional relational boolean expression
union> ElementaryRBE, BooleanAnd


! Boolean or
allOf> disjunction=DisjunctionalRBE
allOf> term=ConjunctionalRBE


! Disjunctional relational boolean expression
union> ConjunctionalRBE, BooleanOr


! Relational boolean expression
union> DisjunctionalRBE


! if { block
allOf> condition=RBE
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> else=ElseBlock
optional> endComment=Comment


! for { block
allOf> counter=Identifier
allOf> range=Range
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> endComment=Comment


allOf> name=TypeName
optional> comment=Comment


allOf> base=TypeSpec


! model { block element
union> Element, EquationsBlock, DropEquationsLine


! List of model { block elements
allOf> element=ModelElement
optional> next=ModelElementLink


! model { block
allOf> name=TypeName
optional> extends=ExtendsSpec
optional> startComment=Comment
optional> elements=ModelElementLink
optional> endComment=Comment


! Specifies that an inlet is optional
nothing>


! Inlet
allOf> name=Identifier
optional> arrayIndex=ArrayIndex


! Definition of an inlet, which may be optional
allOf> inlet=Inlet
```

```
optional> optional=Optional


! List of inlet definitions
allOf> inlet=InletDef
optional> next=InletDefLink


allOf> inlets=InletDefLink
optional> comment=Comment


! Outlet
allOf> name=Identifier
optional> arrayIndex=ArrayIndex


! Definition of an outlet
allOf> outlet=Outlet


! List of outlet definitions
allOf> outlet=OutletDef
optional> next=OutletDefLink


allOf> outlets=OutletDefLink
optional> comment=Comment


! Label for an inlet, outlet or connection
allOf> name=Identifier
allOf> label=TypeName
optional> comment=Comment


union> LabelDef


! labels { block element
union> BlankLine, LabelDef


! List of labels { block elements
allOf> element=LabelsElement
optional> next=LabelsElementLink


! labels { block
optional> startComment=Comment
optional> labels=LabelsElementLink
optional> endComment=Comment


! atomic, composite or flexible unit { block element
union> InletsLine, OutletsLine, LabelLine, LabelsBlock


! atomic unit { block element
union> Element, UnitElement, EquationsBlock, DropEquationsLine


! List of atomic unit { block elements
allOf> element=AtomicUnitElement
optional> next=AtomicUnitElementLink


! atomic unit { block
```

```
optional> name=TypeName
optional> extends=ExtendsSpec
optional> startComment=Comment
optional> elements=AtomicUnitElementLink
optional> endComment=Comment


! Subunit definition
union> VariableDef


union> SubunitDef


! subunits { block element
union> BlankLine, SubunitDef


! List of subunits { block elements
allOf> element=SubunitsElement
optional> next=SubunitsElementLink


! subunits { block
optional> startComment=Comment
optional> subunits=SubunitsElementLink
optional> endComment=Comment


! Inlet export, connects an exported inlet to a subunit's inlet or a
! sink
allOf> external=Inlet
allOf> internal=Reference
optional> name=Identifier
optional> comment=Comment


! Outlet export, connects an exported outlet to a subunit's outlet or a
! source
allOf> external=Outlet
allOf> internal=Reference
optional> name=Identifier
optional> comment=Comment


! Connection from a subunit's outlet or a source to a subunit's inlet or
! a sink
allOf> outlet=Reference
allOf> inlet=Reference
optional> name=Identifier
optional> comment=Comment


! else { block for connections
optional> comment=Comment
optional> equations=ConnectionsElementLink


! if { block for connections
allOf> condition=RBE
optional> startComment=Comment
optional> equations=ConnectionsElementLink
optional> else=ConnElseBlock
optional> endComment=Comment


! for { block for connections
allOf> counter=Identifier
```

```
allOf> range=Range
optional> startComment=Comment
optional> equations=ConnectionsElementLink
optional> endComment=Comment


! connections { block element
union> BlankLine, InletExport, OutletExport, Connection, ConnIfBlock,
ConnForBlock


! List of connections { block elements
allOf> element=ConnectionsElement
optional> next=ConnectionsElementLink


! connections { block
optional> startComment=Comment
optional> connections=ConnectionsElementLink
optional> endComment=Comment


! parameter specifications { block
! Equations specifying parameters (rather than variables), as if the
! parameter
! were passed at the location of the instantiation.
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> endComment=Comment


! specifications { block
! Equations specifying variables. Each equation must refer to only one
! subunit
! (otherwise, use non-local specifications).
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> endComment=Comment


! non-local specifications { block
! Equations, specifying variables, which can refer to multiple subunits.
optional> startComment=Comment
optional> equations=EquationsElementLink
optional> endComment=Comment


! Source
allOf> name=Identifier
optional> arrayIndex=ArrayIndex
optional> equations=RelationLink


! List of sources
allOf> source=Source
optional> next=SourceLink


allOf> sources=SourceLink
optional> comment=Comment


! Sink
allOf> name=Identifier
optional> arrayIndex=ArrayIndex


! List of sinks
```

```
allOf> sink=Sink
optional> next=SinkLink


allOf> sinks=SinkLink
optional> comment=Comment


! composite unit { block element
union> Element, UnitElement, SourcesLine, SinksLine, SubunitLine,
SubunitsBlock, ConnectionsBlock, ParameterSpecificationsBlock,
SpecificationsBlock, NonLocalSpecificationsBlock


! List of composite unit { block elements
allOf> element=CompositeUnitElement
optional> next=CompositeUnitElementLink


! composite unit { block
allOf> name=TypeName
optional> extends=ExtendsSpec
optional> startComment=Comment
optional> elements=CompositeUnitElementLink
optional> endComment=Comment


! Inlet or outlet rename
allOf> external=Identifier
allOf> internal=Identifier


! List of inlet or outlet renames
allOf> inletOutletRename=InletOutletRename
optional> next=InletOutletRenameLink


! Choice of unit implementing the flexible unit
allOf> choice=TypeSpec
optional> inletOutletRenames=InletOutletRenameLink
optional> comment=Comment


! choices { block element
union> BlankLine, Choice


! List of choices { block elements
allOf> element=ChoicesElement
optional> next=ChoicesElementLink


! choices { block
optional> startComment=Comment
optional> choices=ChoicesElementLink
optional> endComment=Comment


! flexible unit { block element
union> BlankLine, UnitElement, ChoicesBlock


! List of flexible unit { block elements
allOf> element=FlexibleUnitElement
optional> next=FlexibleUnitElementLink


! flexible unit { block
```

```
allOf> name=TypeName
optional> extends=ExtendsSpec
optional> startComment=Comment
optional> elements=FlexibleUnitElementLink
optional> endComment=Comment


! process { block element
union> Element, LabelLine, LabelsBlock, SourcesLine, SinksLine,
SubunitLine, SubunitsBlock, ConnectionsBlock,
ParameterSpecificationsBlock, SpecificationsBlock,
NonLocalSpecificationsBlock


! List of process { block elements
allOf> element=ProcessElement
optional> next=ProcessElementLink


! process { block
allOf> name=TypeName
optional> extends=ExtendsSpec
optional> startComment=Comment
optional> elements=ProcessElementLink
optional> endComment=Comment


! Top-level element
union> Element, ImportLine, StreamBlock, ModelBlock, AtomicUnitBlock,
CompositeUnitBlock, FlexibleUnitBlock, ProcessBlock


! Linked list of top-level elements
allOf> element=CPMElement
optional> next=CPMElementLink


! Start category
optional> elements=CPMElementLink
```

References

[1] Ali Baharev, Lubomir Kolev, and Endre Rév. Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation. *AIChE Journal*, 57:1485–1495, 2011.

[2] Ali Baharev and Arnold Neumaier. A globally convergent method for finding all steady-state solutions of distillation columns, 2013. submitted.

[3] Amy R. Ciric and Peizhi Miao. Steady state multiplicities in an ethylene glycol reactive distillation column. *Ind. Eng. Chem. Res.*, 33:2738–2748, 1994.

[4] M. F. Doherty, Z. T. Fidkowski, M. F. Malone, and R. Taylor. *Perry's Chemical Engineers' Handbook*. McGraw-Hill Professional, 8th ed., 2008.

[5] K. Kofler P. Schodl H. Schichl F. Domes, A. Neumaier. *Concise Manual*, 2012.

[6] E.W. Jacobsen and S. Skogestad. Multiple steady states in ideal two-product distillation. *AIChE Journal*, 37:499–511, 1991.

[7] W. K. Lewis and G. L. Matheson. Studies in distillation. *Ind. Eng. Chem.*, 24:494–498, 1932.

[8] K.Kofler F. Domes H. Schichl P. Schodl, A. Neumaier. *Towards a Self-reflective, Context-aware Semantic Representation of Mathematical Specifications*. Springer, 2012.

[9] John M. Prausnitz, Rüdiger N. Lichtenthaler, and Edmundo Gomes de Azevedo. *Molecular Thermodynamics of Fluid-Phase Equilibria*. Prentice Hall PTR, Upper Saddle River, NJ, third ed., 1999.

[10] Buford D. Smith. Design of Equilibrium Stage Processes. In *McGraw-Hill Series in Chemical Engineering*. McGraw-Hill Book Company, Inc., New-York, 1963.

[11] Chang K. Yi and William L. Luyben. Design and control of coupled reactor/column systems–Part 1. A binary coupled reactor/rectifier system. *Computers & Chemical Engineering*, 21(1):25–46, 1996.