**Chemical 2.0**

**(Free open-source Modelica library)**

Marek Mateják1,2

1Institute for Clinical and Experimental Medicine, Czech Republic

2First Faculty of Medicine, Charles University, Czech Republic

marek@matfyz.cz

# **Abstract**

Free open-source Modelica library called Chemical 2.0 (<https://github.com/MarekMatejak/Chemical>) provides robust and unified chemical substance and process definitions (as extension of Modelica.Medial.IdealGases); propagation of definitions and chemical solution state through connected components; working with unknown substances; chemical kinetics; working with variable (exact change calculation such as in previous versions) or simplified constant chemical solution state (unchanged during simulation of chemical processes).   
The possibilities and performance of chemical pathways modeling are increased using a new type of connectors based on inertial electro-chemical potential (as analogy of ThermofluidStream). Chemical processes can be directly connected between them to allow user modeling complex chemical pathways in user friendly way without need to identify unsignificant substances accumulations. Chemical reactions can be parametrized also in common way using forward rate coefficient and dissociation coefficient.

Keywords: Chemical 2.0, Modelica library, physical chemistry, thermodynamics equilibria, electrochemical potential, electrochemical cell, internal energy, semipermeable membrane, chemical kinetics, chemical pathways

# **Introduction**

Motivation to improve Chemical library from version 1.4 to version 2.0 comes from following Modelica libraries:

1. Modelica.Media.IdealGases: Chemical substance and chemical process can be described by the selected extensive physical quantities.
2. ThermofluidStream: Inertial connectors can be generalized for chemical domain.

As selection of physical quantities for substance and process definition we chose similar approach as Modelica.Media.IdealGases, which comes from [McBride B.J., Zehe M.J., and Gordon S. (2002): NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. NASA report TP-2002-211556]. Having this definition as a record of constant coefficients, there is a possible to calculate molar heat capacity, free molar formation energies, free molar entropy and many other thermodynamic properties in dependence on current substance activity and current state of chemical solution (temperature, pressure, electric potential, …). And more, there is no more needed to have separate calculations for different state of matters (phases) as in previous version because phase is also included in extensive physical quantities of the definition records. As a result, each chemical equilibrium of each process can be expressed only using these definitions of included substances and processes. And because we chose extensive quantities, they can be easily summed or extracted as common chemical notations of chemical processes (e.g. mass of substrates = mass of products; charge of substrates = charge of products; enthalpy of substrates + enthalpy of reaction = enthalpy of products, entropy of substrates + entropy of reaction = entropy of products, …).

The connectors from ThermofluidStream library are revolutionary. Library still allows user to define very complex thermodynamic behavior of predefined media in the same level of detail as in Modelica.Fluid and Modelica.Media does. However, ThermofluidStream library allows to connect processes without unnecessary accumulations of media in the same time as it eliminates number of non-linear systems in the model. This makes model more readable, because there is no need to have many middle-state components accumulating and mixing medium. And during run of the simulation there is no more needed to solve implicit equations for many non-linear systems. Performance increases rapidly with better readability without minimal compromise with modeled details of the system.

Using the inertial approach of ThermofluidStream in chemical domain promises the analogical improvements for Chemical 2.0. As a result, the processes can be connected directly together allowing to define complex chemical pathways such as in biochemistry or physiology. And at the same time, it allows to define better chemical kinetics, complex processes and more detailed calculations of any part of the model.

# **Definition**

The previous version of Chemical library has various definitions of substances. In version 2.0, all these structures for all type of substances and even for all type of chemical processes are unified into just one type of operator record called Chemical.Interfaces.Definition. This record can define each substance and process used in the previous version. In addition, the user can set their values independently on internal representation using suitable constructors or function processData. Function processData creates process definition from dissociation constant (molar-based) and consumed heat of the process (free molar enthalpy change). Using relation between this process definition and its substances definitions, the user can easily evaluate new definition of new chemical substance or new chemical process using intuitive ‘\*’, ‘+’ and ‘-‘ operators.

For example, a definition of aqueous O2 can be set from tabulated Henry’s coefficient (0.0013, 1500K). For another example, a definition of H2O formation reaction from H2 and O2 can be set as an algebraic equation between reaction products and substrates (**Listing 1**).

**Listing 1.** Example of definitions

import Chemical.Interfaces.Definition;

import Chemical.Substances.Gas;

import Chemical.Substances.Liquid;

import Chemical.Interfaces.processData;  
constant Real R = Modelica.Constants.R;

constant Definition **O2\_aq** =

Gas.O2 + processData(

K = 0.0013,

dH = -1500\*R);

constant Definition **H2O\_formation** =

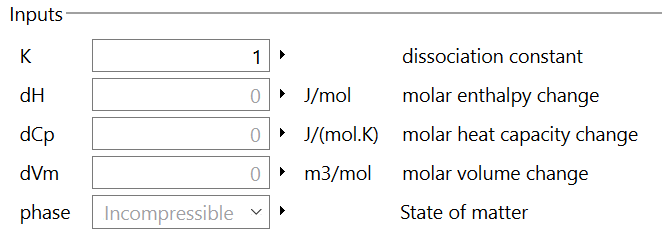
Gas.H2O - (Gas.H2 + 0.5\*Gas.O2);

constant Definition **Hemoglobin** =

Liquid.Unknown;

The default definition of Unknown substance is molar mass 1kg/mol and molar heat capacity 1 J/(mol.K) with zero free formation Gibbs energy and zero free formation enthalpy. These values are used if the user does not set the substance. Because free formation energies are always defined in a relative way, unknown substances can be used such as any base substance with unknown formation process. If they play a role in solution properties, at least the adequate property should be set. E.g. molar mass and molar volume must be set if they play a role in solution composition; molar heat capacity must be set if it plays a role in heat accumulation in chemical solution.

**Figure 1**, processData() inputs

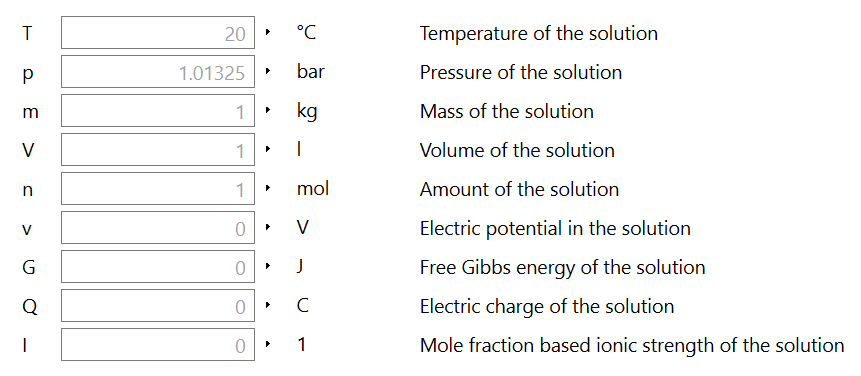


# **Chemical solution**

There are two options for how to define a chemical solution. The first is using the Chemical.Solution component and connecting all substances through its solution connector. This was the only option in the previous version, and it works in the same way also in this version. If a chemical solution is defined this way, then all its properties are dynamically calculated during simulation.

However, there is not always a need to change the solution properties. If solution properties, such as temperature, pressure, total mass, total volume, total amount of particles, and electric potential, can be assumed as constant during simulation, then a solution state can be set as parameter. If the user did not set a solution property, (**Listing 2**) then its default value (**Figure 2.** Default chemical solution) will be used.

**Figure 2**. Default chemical solution properties



**Listing 2.** Example of solution

import Chemical.Interfaces.SolutionState;

import Chemical.Interfaces.Phase;

constant SolutionState **SATP** =

SolutionState(phase = Phase.Gas,

T = 298.15);

SolutionState **heatingSolution** =

SolutionState(phase = Phase.Gas,

T = 273.15+time);

# **Properties of chemical processes**

Each process has different properties (e.g. dissociation coefficient, free Gibbs energy or free enthalpy) in dependence on its definition and on current state of chemical solution (e.g. temperature, pressure, …). The user needs to connect the substance definition and solution to model Chemical.Interfaces.ProcessProperties to evaluate these properties.

**Listing 3.** Example of process properties

import Interfaces.ProcessProperties;

ProcessProperties **O2\_dissolving\_props**

(definition = processData(0.0013,-1500\*R),

solutionState = heatingSolution);

ProcessProperties **H2O\_formation\_props**

(definition = H2O\_formation,

solutionState = heatingSolution);

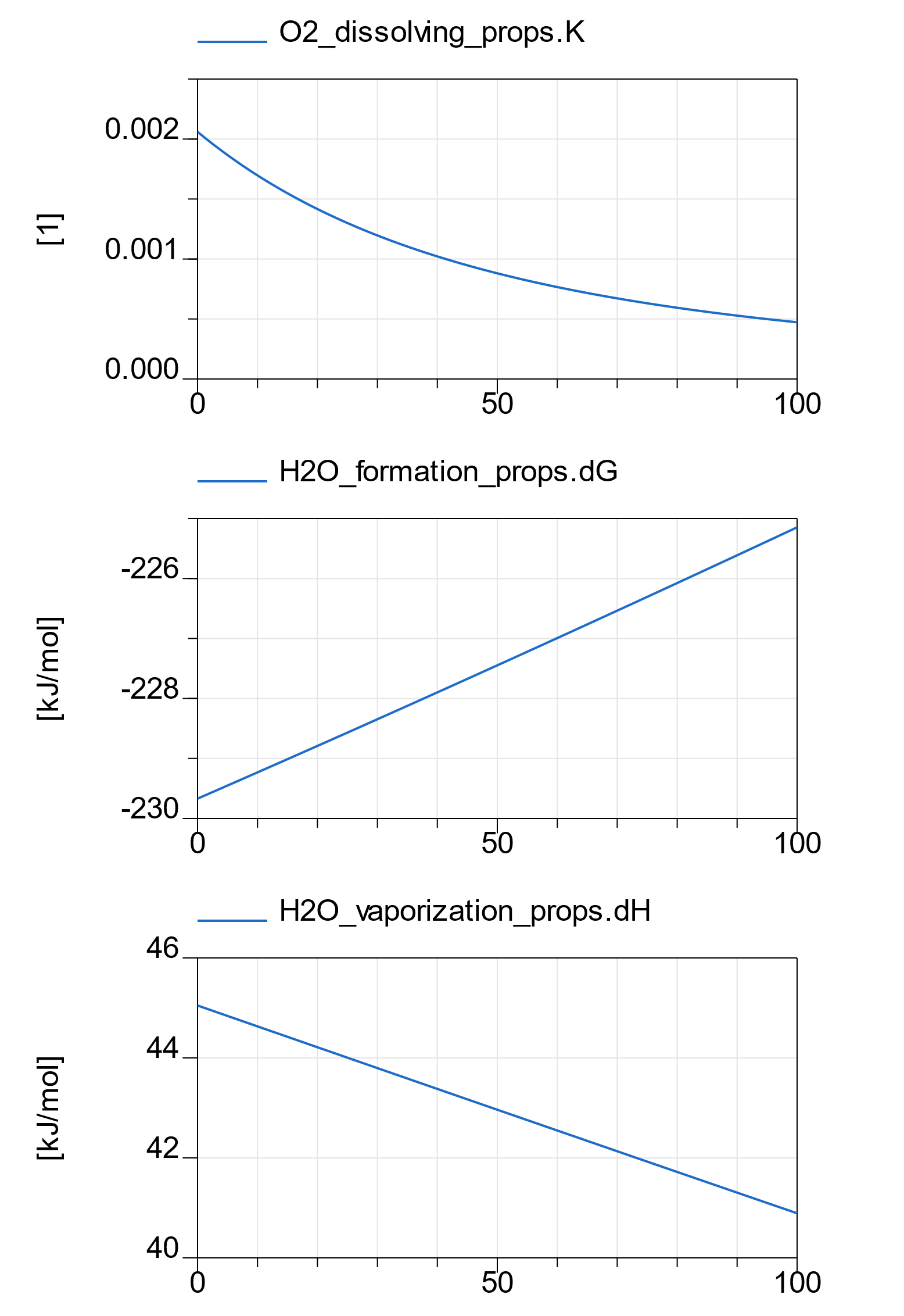
ProcessProperties **H2O\_vaporization\_props**

(definition = Gas.H2O - Liquid.H2O,

solutionState = heatingSolution);

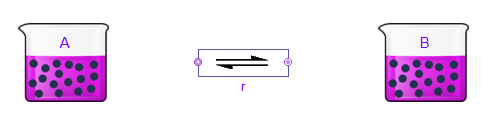
After having run the simulation for 100 seconds, the temperature of the solution changed from 0°C to 100°C, and we can see the recalculated properties that resulted from this temperature change e.g. dissociation constants (Henry’s coefficient), free Gibbs energy change of reaction or consumed heat by process (free enthalpy change) – e.g. **Figure** 3, Example of process properties.

**Figure 3**, Example of process properties

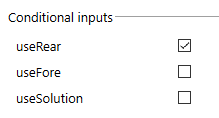


# **Chemical process**

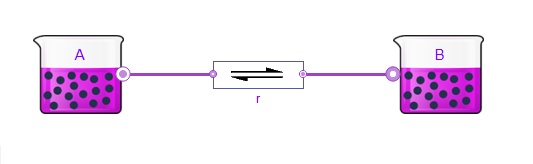
To create a model of a chemical reaction using Chemical 2.0, drag and drop substance component Substance from package Chemical.Boundaries as substance A and substance B. In the same way, put into the model Reaction component r from package Chemical.Processes.



The default setting for the substances does not provide any connector, so it is necessary to setup which connectors will be used. This is done in parameter dialog by clicking the useFore for substance A and useRear for substance B.



Having all necessary connectors, it is possible to connect substances with chemical reaction r.



This model could be simulated, but it all remains constant because the dissociation coefficient is equal to one and both substances are the same (defined as Liquid.Unknown) with amount of substance 1mmol by default. However, if we change the dissociation constant to 2 (resulting in code **Listing 4.**) then we can see dynamic behavior in constant default chemical solution. And after 5s of simulation time, the reaction reaches almost the equilibrium where the amount of product B is two times higher than the amount of substrate A. We can change reaction speed by setting forward reaction rate coefficient (r.k\_forward) lower than its default value. Backward reaction rate is not possible to set as a reaction parameter because it is defined as K\*k\_forward. This reaction can have as many substrates (nS) and as many products (nP) as needed, and its index is defined by in order of drown connection. Note, that default chemical solution in this example is set to **Figure 2** and remains constant during simulation (incompressible at 20°C, 1atm with standard molality 1 mol/kg, standard molarity 1 mol/L).

**Listing 4.** Generated code for a reaction example

model SimpleReaction

 extends Modelica.Icons.Example;

import Chemical.Boundaries.Substance;

import Chemical.Processes.Reaction;

import Chemical.Interfaces.processData;

Substance **A**( useFore = true );

Substance **B**( useRear = true );

Reaction **r**( nP = 1, nS = 1,

process = processData(2) );

equation

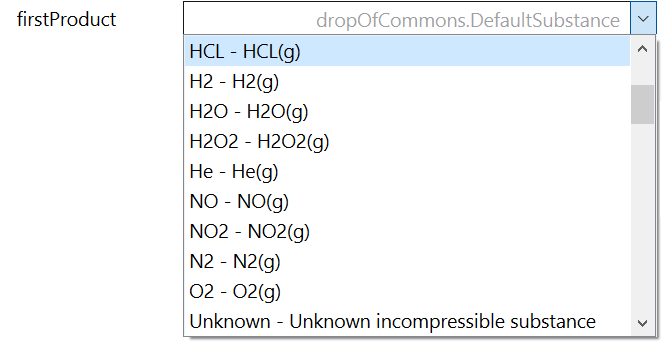
  connect( A.fore , r.substrates[1] );

  connect( r.products[1] , B.rear );

end SimpleReaction;

If the user prefers first product definition for a chemical process, then it is possible to set process option “frstProductFrom” into “Substance” selection. This enables the parameter firstProduct (the first product definition) in the same dialog of the process setting. Because with knowledge of all products and all substrate definitions the process definition can be evaluated, so the parameter with process definition is disabled in this selection.

# 



In addition, stoichiometric coefficients can be set as parameter s for substrates and parameter p for products as usual in previous version. The new extension for processes allows to change chemical kinetics as redefinition of relation between electro-chemical gradient and process molar rate called uLoss. By default is uLoss set to traditional chemical kinetics based on forward rate coefficient, but user can define own type of chemical kinetics constrained by proposed interface for potential loss functions.

**Listing 4.** Example of reaction defined by product with uncommon chemical kinetics

import Chemical.Utilities.Types.

FirstProductChoice;

import Chemical.Processes.Internal.Kinetics;

Chemical.Processes.Reaction **H2\_burning** (

firstProductFrom =

FirstProductChoice.Substance,

    firstProduct = Gas.H2O,

nS=2, s = {2,1},

    nP=1, p = {2},

    redeclare function uLoss =

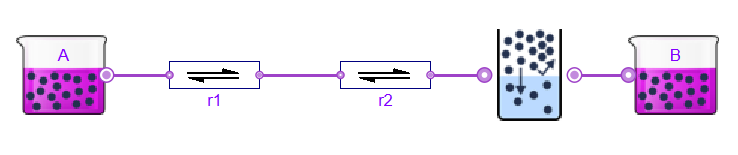
Kinetics.fastPotentialLoss,

)

# **Data propagation**

The version 2.0 of the Chemical library is based on a special type of connector firstly defined by the ThermofluidStream library. These connectors propagate the state of media without inertial part in flow direction and generate equality of inertial electro-chemical potential instead of direct equality of the total electro-chemical potential. The equilibria and even the dynamic simulation reach almost the same values if inertia is fast enough. As proved by …… this type of connector has physical background and better performance because it eliminates non-linear systems of equations where solvers typically spend the most of time. Moreover, this design allows users to connect more processes together without any accumulation (or just with a few) of substance. As a result, it is possible to define long chemical pathways as are usual in physiology.

**Figure 4**, Example of possible pathway connection



To have a chemical pathway such as in **Figure *4*** there is a need to have each substance definition between each chemical process. This is done automatically using a propagation pattern from forward connectors (called fore) into rearwards connectors (called rear). Please note that substances definition propagation is not dependent on flow direction. Each substance definition remains constant during simulation, so the direction of definitions propagation does not take any role during simulation and vice versa. In **Figure *4*** only the substance A can be defined. The definition of a product of reaction r1 is evaluated from the definition of this reaction and its substrates. So generally, each process can define the first product from its other products, substrates, and process definition. Even substance B must not be explicitly defined because its definition can come from PhaseTransition process.

In the same way, the constant state of the chemical solution can be propagated. If the state of chemical solution is constant, it remains the same for each substance in the solution during each step of the simulation. In the model shown in **Figure *4*** it means that chemical solution is specified only in substance A and propagated trough connections to each other components as the same constant values.

To model the situation when processes cross different solutions, there is an option to break the solution propagation. In some processes, such as Diffusion, Membrane or PhaseTransition, the solution states are not propagated by default because they change the chemical solution between substrates and products. As a result, the new solution must be defined (if are not sufficient with default solution **Figure 5**) for the process and each its product to propagate into next processes. Please note, that “next” is here not defined by molar flow but defined by fore-rear connections. If the kinetics is dependent on

**Listing 5.** Example of process changing chemical solution

import Chemical.Processes.PhaseTransition;

import Chemical.Interfaces.SolutionState;

import Chemical.Interfaces.Phase;

import Chemical.Substances.Gas;

PhaseTransition PhaseTransition(

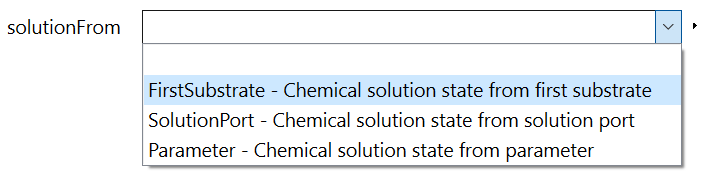
     solutionParam =

SolutionState( phase = Phase.Gas ),

     Product = Gas.CO2 );

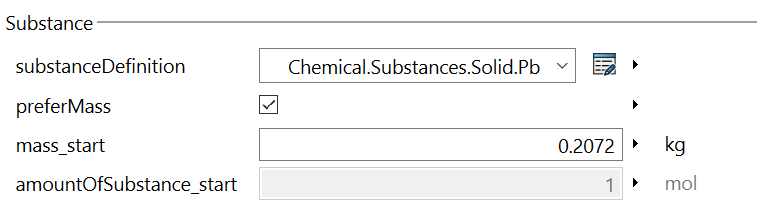
There is no difference difference between Diffusion and Membrane process. And only difference between Diffusion ang PhaseTransition component is that during Diffusion the substrate remains the same as product. This allows to use PhaseTransition even for solvation (gas dissolution in liquids, gas volatility from liquids) or for phase transition (vaporization, condensation, sublimation, deposition, melting, freezing). Note that the definition of chemical each compound is dependent on phase and on solvent. For example, gas water was different definition as liquid water. Even the phase transition as a process consumes/ releases heat during this process.

Also, if a chemical substance significantly changes the chemical solution properties, then it must be connected to Solution component using SolutionPort.



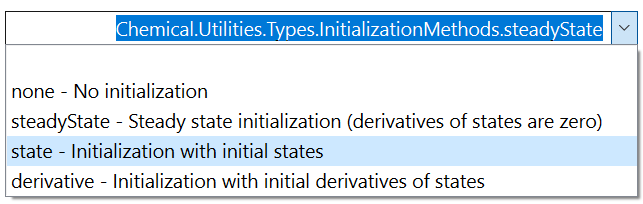
# **Initialization**

Substance component is initialized by initial mass or amount of base substance molecules. Recalculation from number of base molecules and mass is trough molar mass, which is part of the substance definition. However, because the substance definition can be propagated from other components the molar mass must not be known during initialization. Internally it is solved by switching between mass-based and molar-based accumulations in dependence on which quantity is selected for initialization using checkbox “preferMass”. If mass is preferred, anountOfSubstance\_start is disabled and mass\_start is enabled as an initial value of substance accumulation.



A principle of the inertial connectors affects the initialization of the model processes. Having acceleration of flows on each connection, each flow has a state. So, the simplest way of initialization is to set values of molar flows. From these values are expressed gradients of electro-chemical potentials on process components as a relations of substance states between boundaries.

Molar flow initialization



Another option is to set initial process state to steady state. This will take inertial electrochemical potential change **r** to zero. If each process has this type of initialization, then the solver is looking for the non-accelerated state of whole system, where derivation of each flow is zero.

# **Discussion**

There is a conviction that complex things could have better usability, better performance, better details and better interfaces then the simplified one. We hope that the step of Chemical library from version 1.4 to version 2.0 has this kind of improvements.

Even the code of the library is not a simple one and there is a lot of physical equations and parameters inside, the user interface and usability is simplified as much as possible. So, you can start with simple examples without almost any settings of components. And after some iterations you could look at assumptions of chemical solutions and settings which are hidden as default values in many cases. This approach allows you to start with very simple model and extending significant details during each modeling iteration.

Theory in the level of physical chemistry is summarized in documentation at Chemical\Resources\Documentation\Chemical.pdf.

# **Acknowledgements**

Special thanks are given to Taras (Terry) Yavorskyy and Dr. Jason Brown, whose made English prove reading of this contribution.

# **References**

TODO