Material system

DuMu^x

Material system

Challenges in simulating porous media:

- Highly heterogeneous distribution of parameters and complex nonlinear material laws.
- Strong interconnection of properties ➡ difficult to achieve modularity

Modular structure by separating the material system into the following parts:

- Components
- Fluid system
- Solid system
- Binary coefficients
- Fluid-matrix interactions
- Spatial parameters

} User-defined parameters and relationships describing the thermodynamical behavior of the system

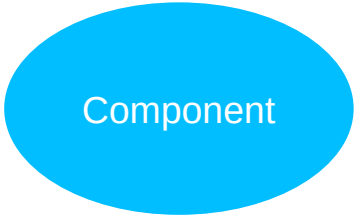
- Fluid states
- Solid states
- Constraint solvers

} Dumux-specific containers and solvers

Material system: Component

What it does:

- **Thermodynamic relations** (e.g. molar mass, vapor pressure, density) of a **single chemical species** or a fixed mixture of species
- Provide a convenient way to access these quantities.
- Not supposed to be used by models directly.



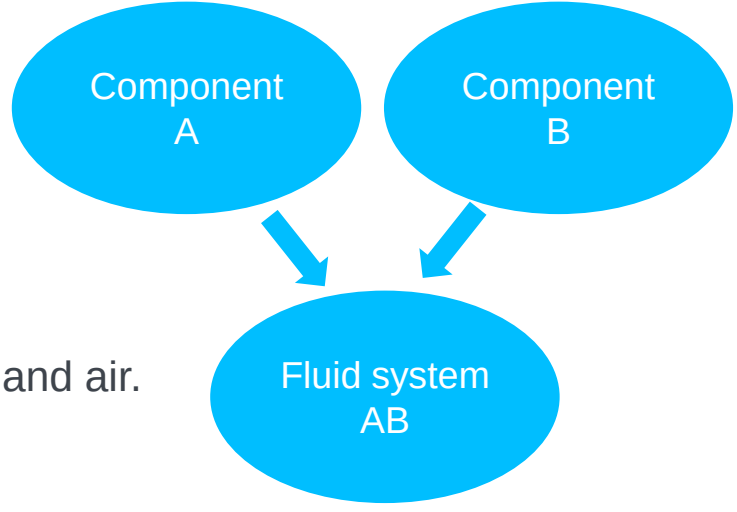
Example implementations:

- `H2O`: pure water, properties by IAPWS-97 or simplified.
- `Brine`: water with a given salt concentration.

Material system: Fluid system

What it does:

- Expresses the **thermodynamic relations between fluid quantities** (e.g. calculation of density or viscosity based on composition, fugacity coefficient based on temperature and pressure…).
- Stateless classes (all member functions are static).



Example implementations:

- `TwoPImmiscible`: two immiscible fluid phases.
- `H2OAir`: gas and liquid phase with components water and air.

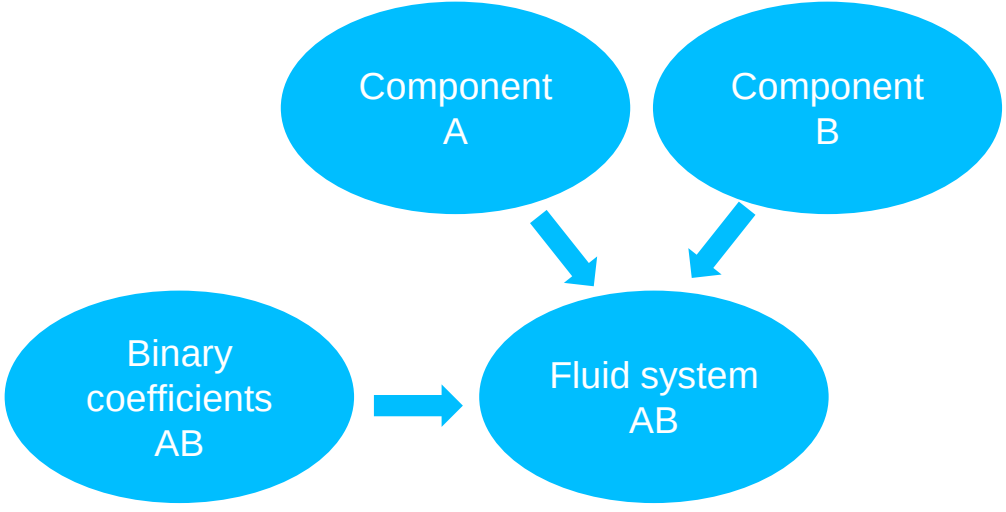
Material system: Binary coefficients

What it does:

- **Contains** data required for binary mixtures (e.g. binary diffusion coefficients, coefficients needed for constitutive relationships (like Henry coefficient))

Example implementations:

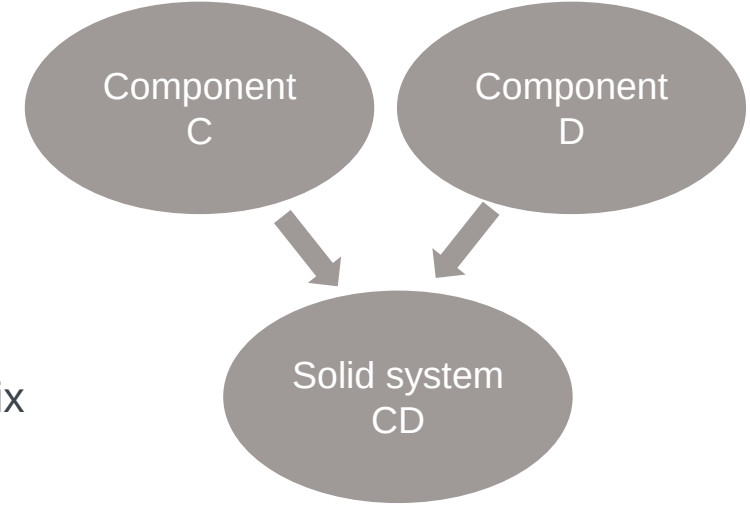
- `H2O_Air`
- `Brine_Air`



Material system: Solid system

What it does:

- Expresses the **thermodynamic properties of the solid matrix** (e.g. calculation of the solid density and solid heat capacity based on the composition…).
- Stateless classes (all member functions are static).



Implementations:

- `InertSolidPhase`: inert solid matrix of one solid component (e.g. granite)
- `CompositionalSolidPhase`: composed solid matrix of inert or reactive components (e.g. NaCl and granite)

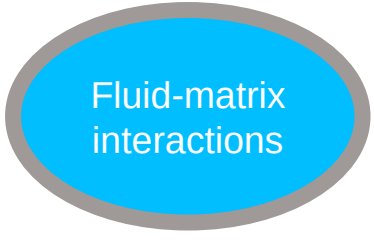
Material system: Fluid-matrix interactions

What it does:

- Description of the **interaction of the fluid phases with the porous medium** (e.g. capillary pressure and relative permeability relationships).
- Through modular adapters, regularization schemes can be imposed for extreme values.

Example implementations:

- `VanGenuchten`
- `BrooksCorey`



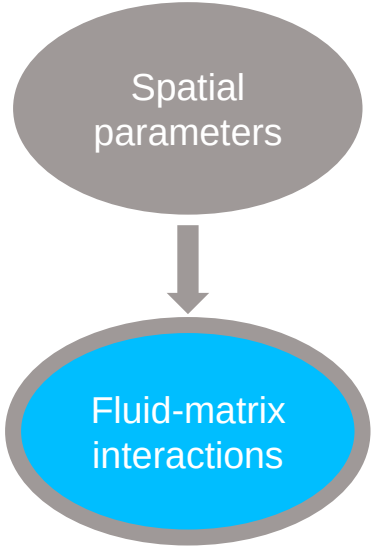
Material system: Spatial parameters

What it does:

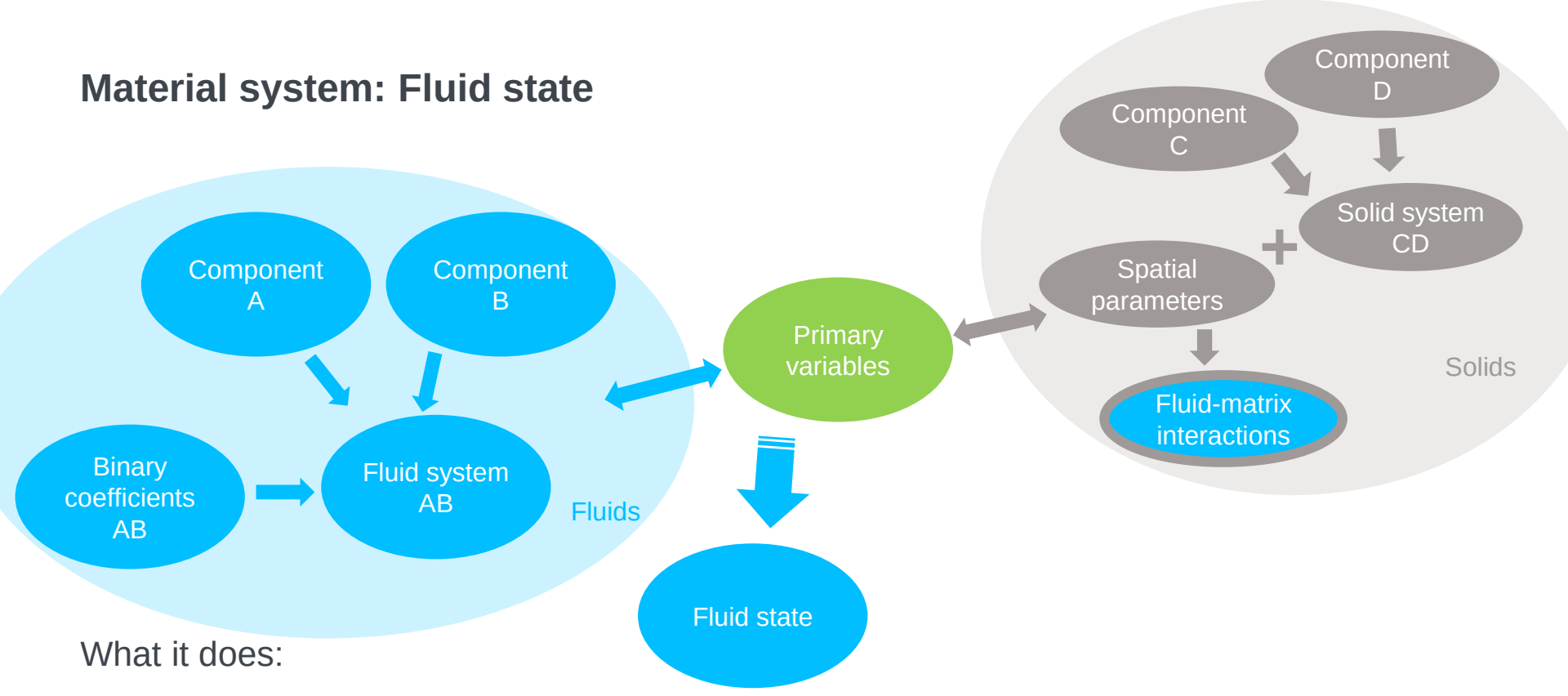
- **Collects** all parameters that may vary **depending on the location** within the porous medium (e.g. permeability, residual saturations)

Example implementations:

- Problem-specific!



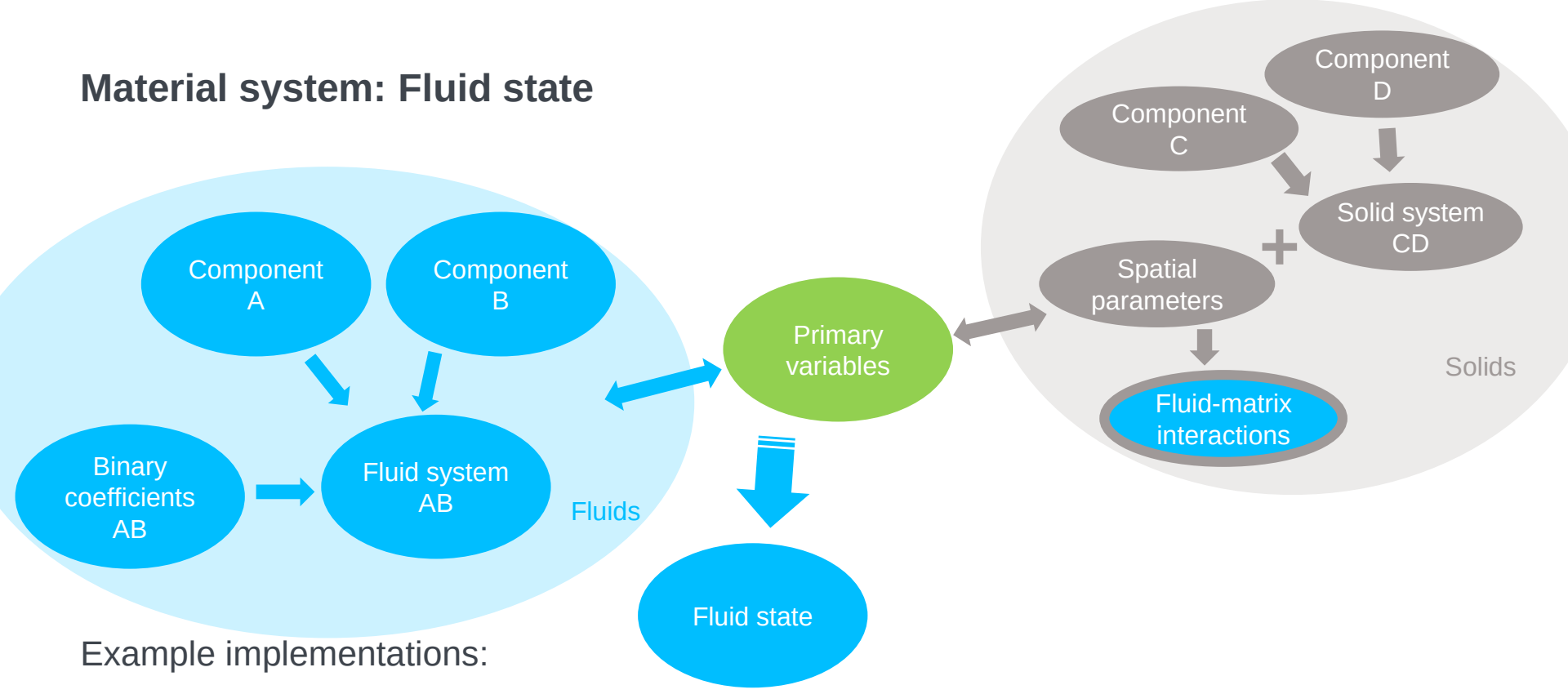
Material system: Fluid state



What it does:

- **Stores** the complete thermodynamic configuration of a system at a given spatial and temporal position (e.g. saturation, mole fraction, enthalpy).
- **Provides access** methods to all thermodynamic quantities (e.g. saturation, mole fraction, enthalpy).

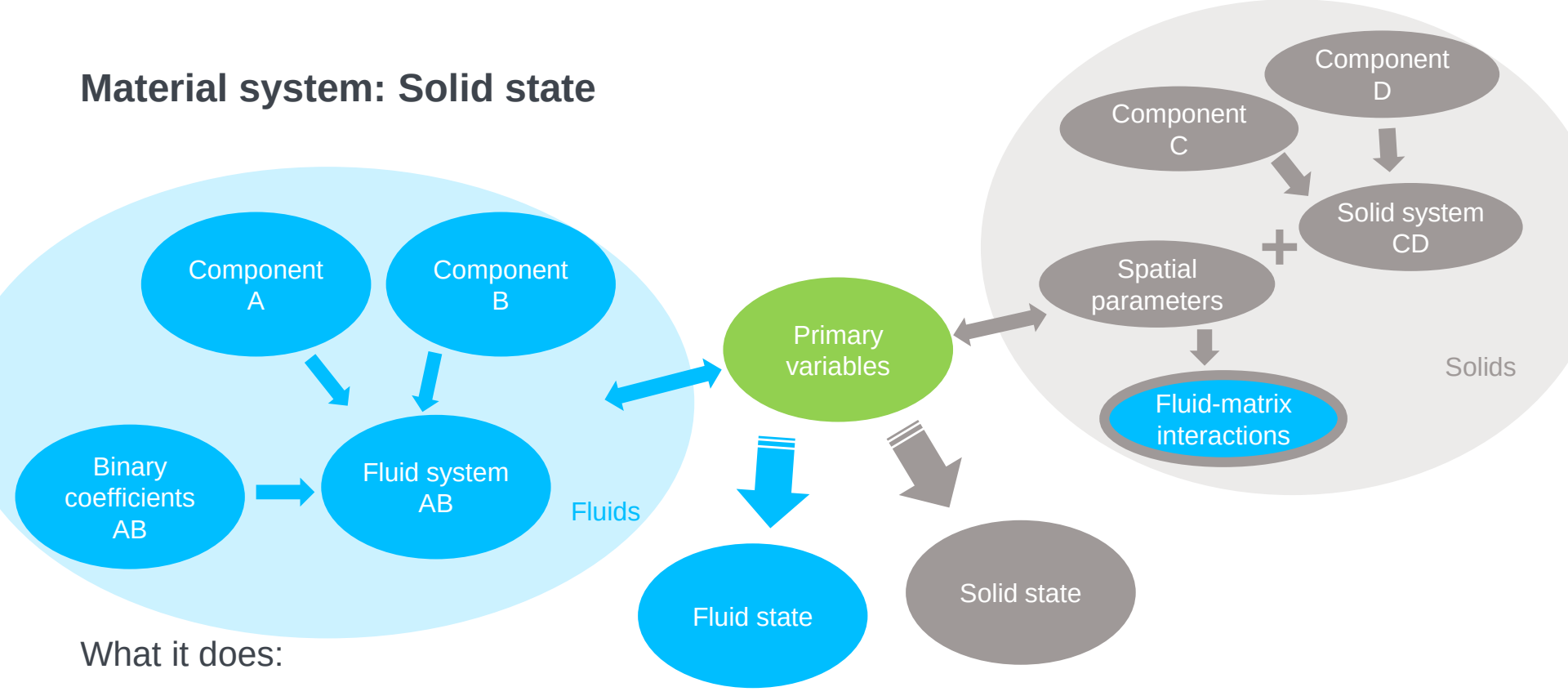
Material system: Fluid state



Example implementations:

- `ImmiscibleFluidState`: assumes immiscibility of the fluid phases. Phase compositions and fugacity coefficients do not need to be stored explicitly.
- `CompositionalFluidState`: assumes thermodynamic equilibrium, only a single temperature needs to be stored.

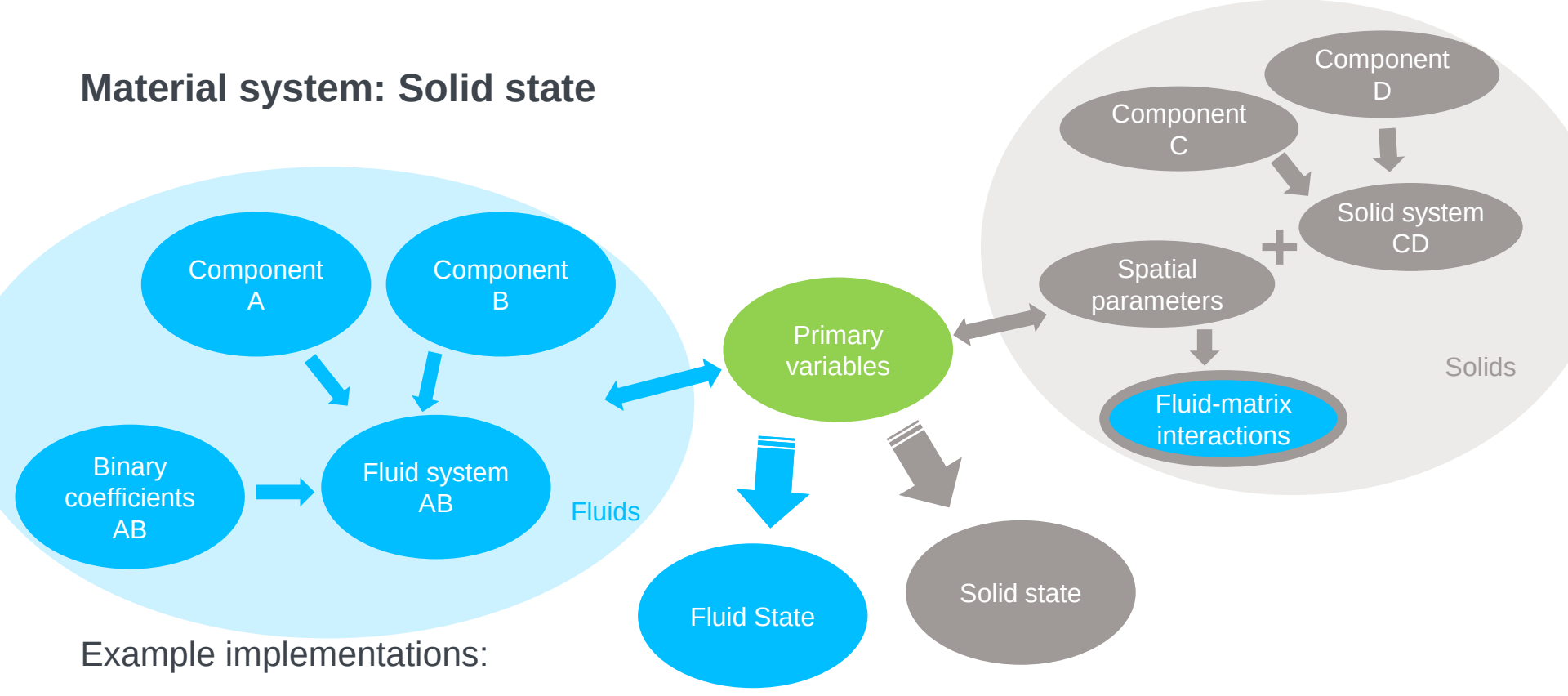
Material system: Solid state



What it does:

- **Stores** the complete solid configuration of a system at a given spatial and temporal position (e.g. solid volume fractions, solid heat capacity).
- **Provides access** methods to all solid quantities (e.g. porosity, density, temperature).

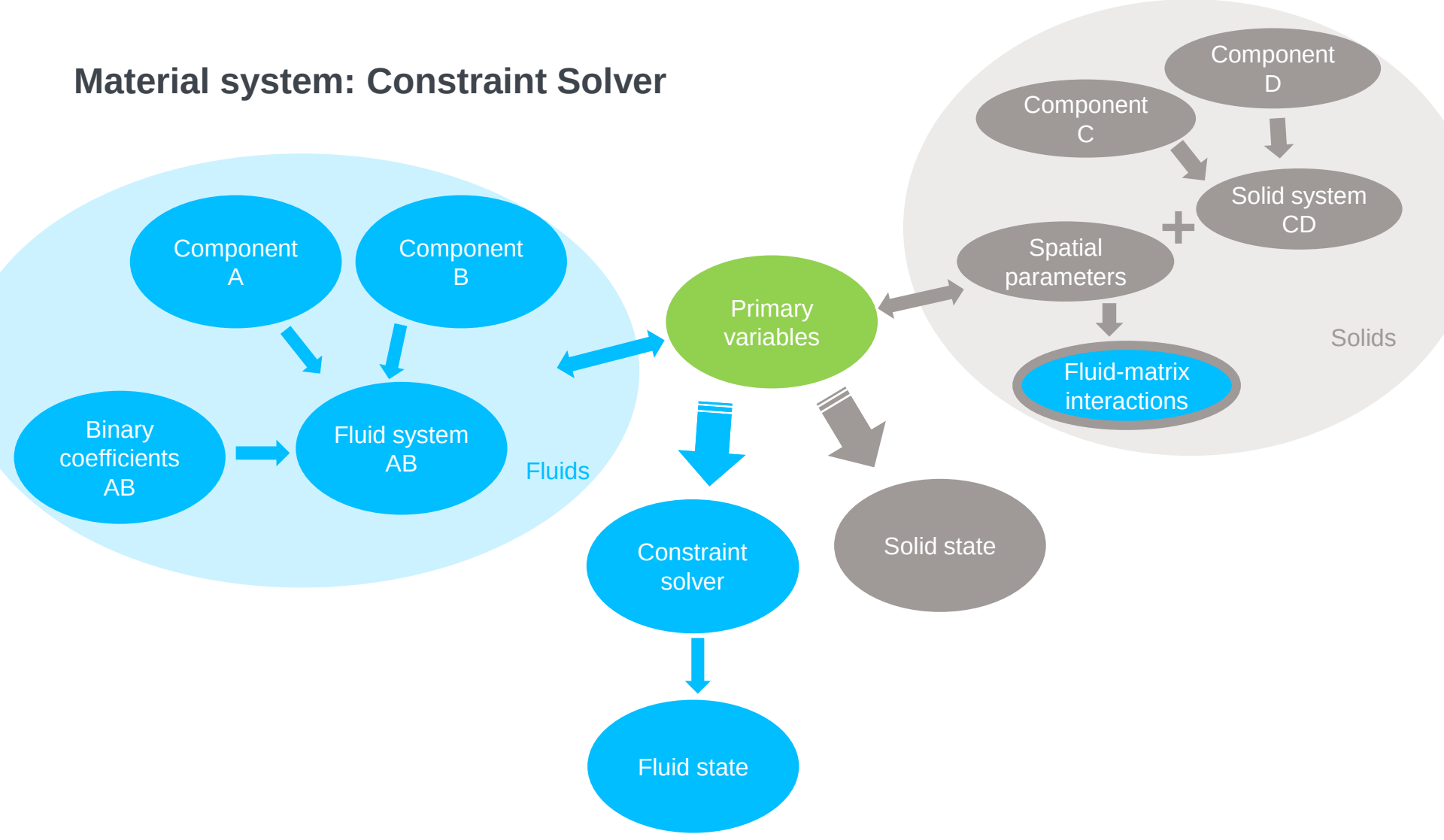
Material system: Solid state



Example implementations:

- `InertSolidState`: assumes an inert solid phase. Solid volume fractions do not change.
- `CompositionalSolidState`: assumes a solid matrix composed out of two components. The volume fractions can change and properties such as heat capacity are adapted.

Material system: Constraint Solver



Material system: Constraint solver

What it does:

- **Connects** the thermodynamic relations expressed by fluid systems with the thermodynamic quantities stored by fluid states (e.g. mole fraction, density).

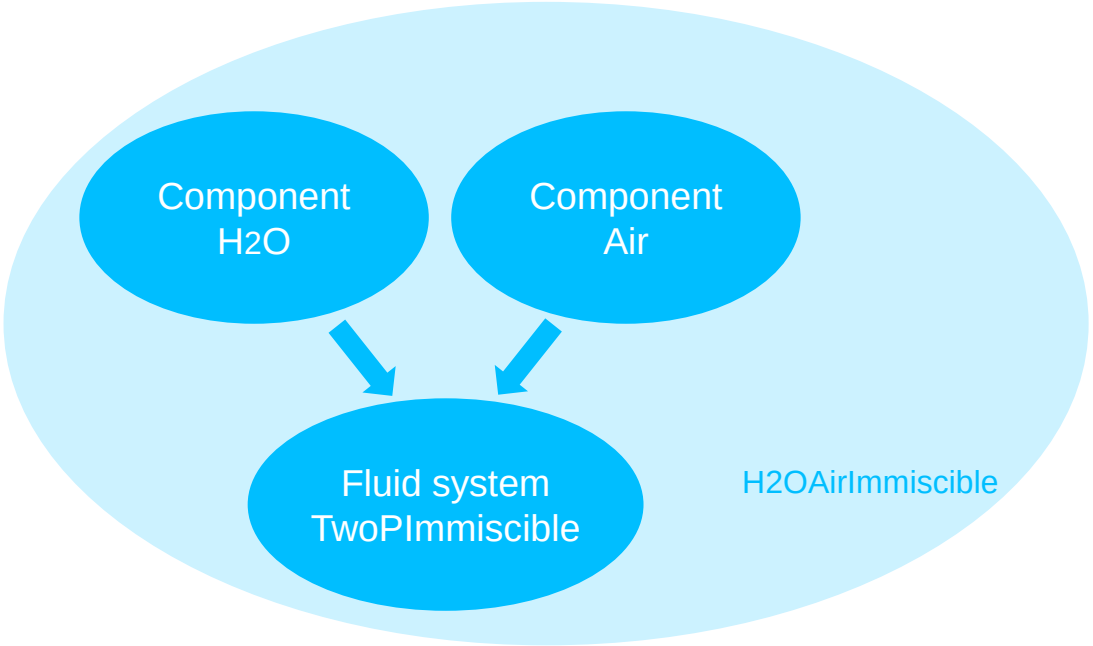
Example implementation:

- `CompositionFromFugacities`: takes all component fugacities, the temperature and pressure of a phase as input and calculates the phase composition.

Example: From component to fluid system

Example: 2 phases, immiscible

- Components: `H2O, Air`
- Fluid system: `TwoPImmiscible`



Example: From component to fluid system

Include headers in problem file:

```
// The numerical model
#include <dumux/porousmediumflow/2p/implicit/model.hh>

// The water component
#include <dumux/material/components/h2o.hh>

// The air component
#include <dumux/material/components/air.hh>

// The two-phase immiscible fluid system
#include <dumux/material/fluidsystems/2pimmiscible.hh>

// Liquid phase
#include <dumux/material/fluidsystems/liquidphase.hh>

// Gas phase
#include <dumux/material/fluidsystems/gasphase.hh>

// Solid phase
#include <dumux/material/solidsystems/inertsolidphase.hh>
```

Example: From component to fluid system

Specify fluid system in problem file:

```
// we use the immiscible fluid system here
SET_PROP(WaterAirProblem, FluidSystem)
{
private:
    using Scalar = typename GET_PROP_TYPE(TypeTag, Scalar);
    using LiquidPhase = typename FluidSystems::LiquidPhase<Scalar, H2O<Scalar> >;
    using GasPhase = typename FluidSystems::GasPhase<Scalar, Air<Scalar> >;

public:
    using type =  typename FluidSystems::TwoPImmiscible<Scalar, LiquidPhase,
                  GasPhase> ;
};
```

Example: From component to solid system

Specify solid system in problem file:

```
// we use the inert solid system here
SET_PROP(WaterAirProblem, SolidSystem)
{
private:
       using Scalar = typename GET_PROP_TYPE(TypeTag, Scalar);
       using ComponentT = Components::Granite<Scalar>;

public:
       using type = SolidSystems::InertSolidphase<Scalar, ComponentT>;
};
```

Note: Specifying a solid system is only necessary if you work with a non-isothermal or mineralization model. If no solid system is specified in the problem file, the default is the inert solid phase with the constant component. For the constant component you can set properties in the input file.

Exercise

Tasks:

1. Get familiar with the code
2. 2p model: Implement a new component
   - 2.1: Incompressible component
   - 2.2: Compressible component
3. 2p2c model: Implement a new fluid system
4. Change wettability of the porous medium
5. Advanced: Use van Genuchten relationship with parameters: alpha = 0.0037 and $alpha_{lense}$ = 0.00045, n = 4.7 and n_{lense} = 7.3

First step: Go to https://git.iws.uni-stuttgart.de/dumux-repositories/dumux-course/tree/master/exercises/exercise-fluidsystem and check out the README

Thank you!