On the Algebraic Properties of Concrete Solution Aggregation

Research

University of Stuttgart

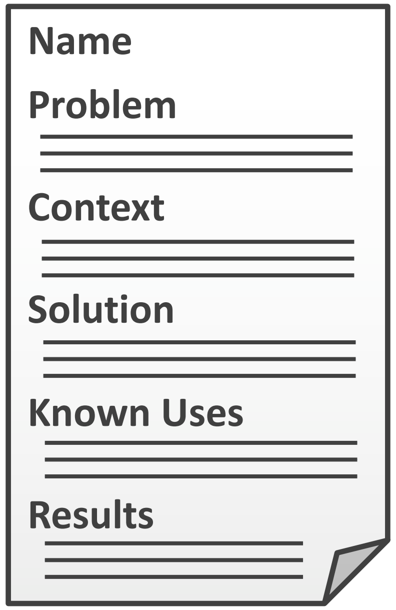
**Michael Falkenthal, Johanna Barzen,
Uwe Breitenbücher, and Frank Leymann**
falkenthal@iaas.uni-stuttgart.de
Institute of Architecture of Application Systems

Outline

- Background on Patterns and Pattern Languages
- Motivation and Problem
- Solution Algebras
- Some Examples of Aggregation Operators
- Conclusion and Future Work

Background: Patterns

- Patterns are used to capture **proven solution knowledge to recurring problems in a human readable way**



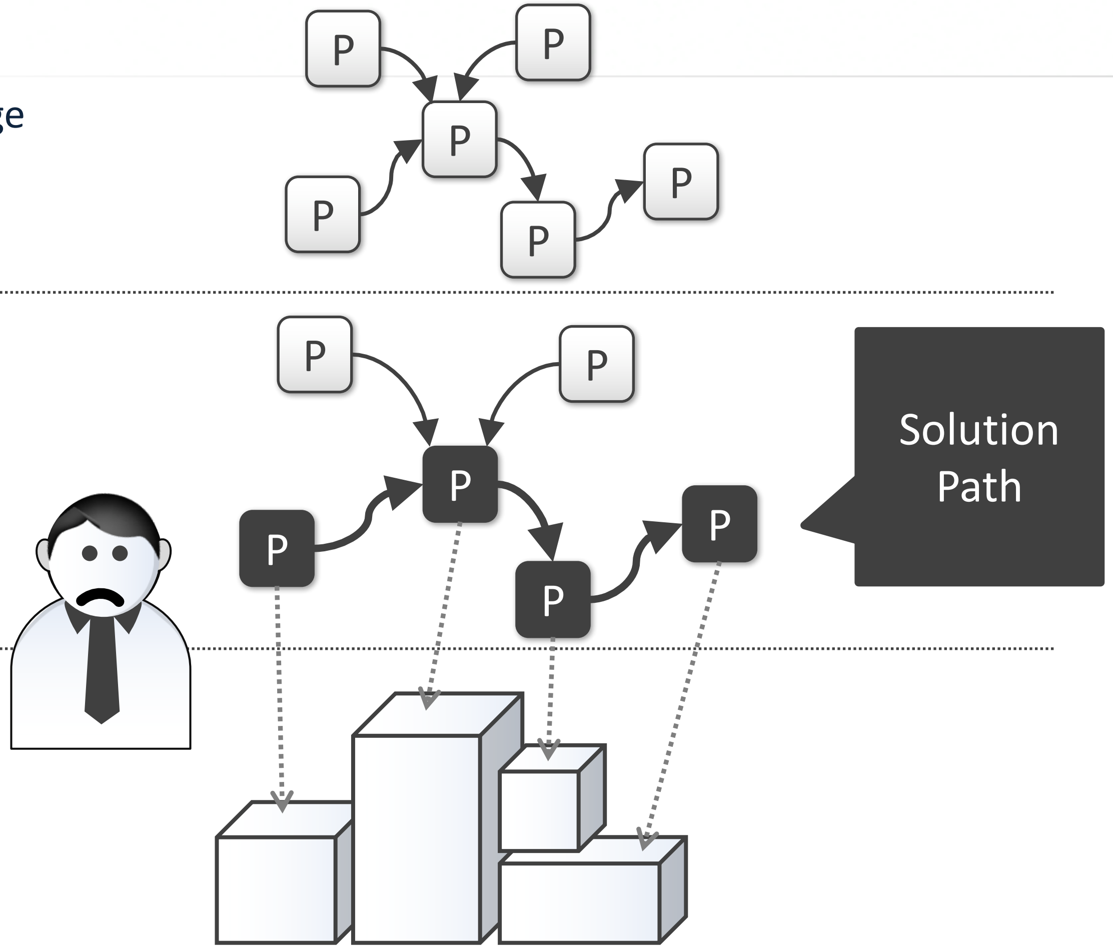
- Patterns aim at **generalization and abstraction** of solution knowledge

Pattern Application Today

1 Study preserved knowledge in *Pattern Language*

2 Selection of *Solution Path* to solve problem at hand

3 *Ad hoc Implementation* of Concrete Solutions and their aggregation



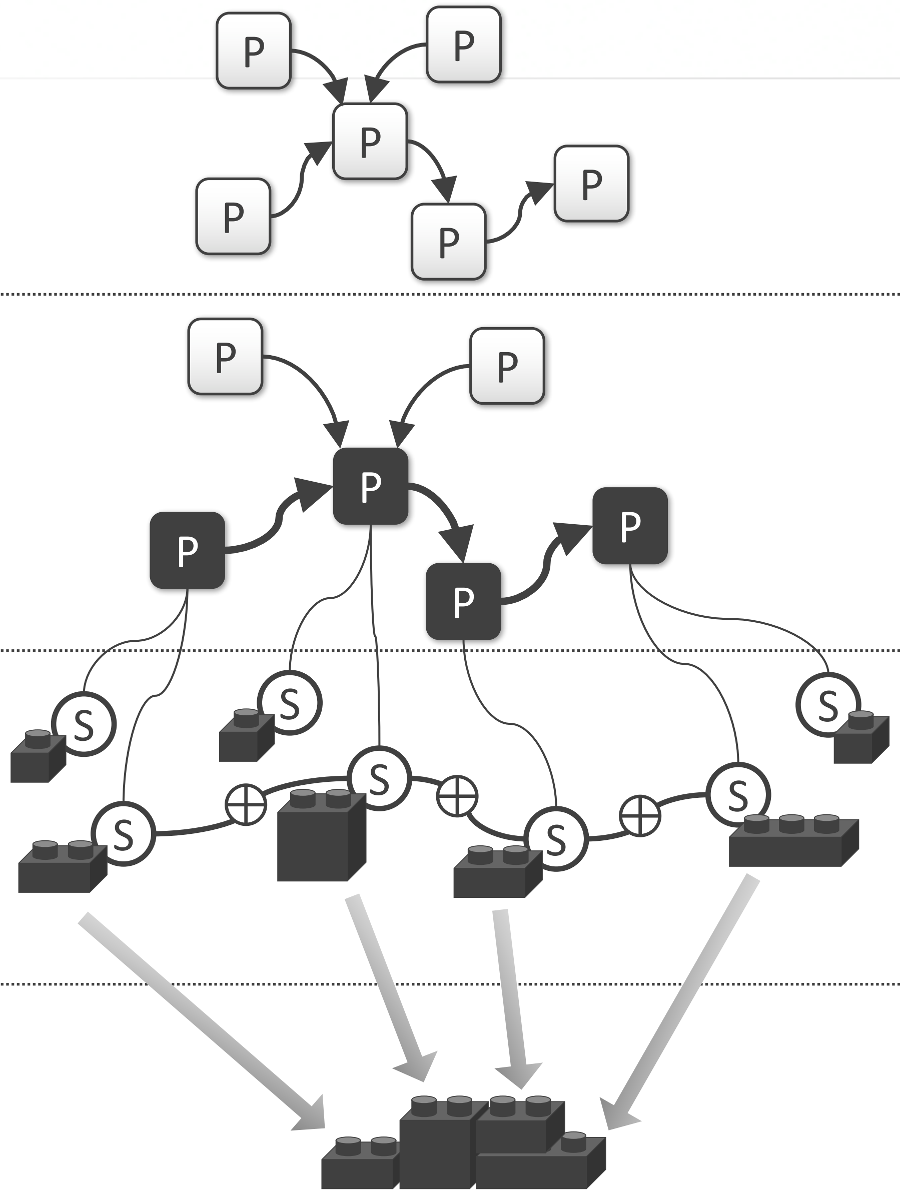
Pattern Application Vision

1. Study preserved knowledge in *Pattern Language*

2. Selection of *Solution Path* to solve problem at hand

3. Selection of *Concrete Solution Path*

4. Generation of *Aggregated Concrete Solution*



How to organize concrete solutions and aggregation operators?

Solution Algebras!

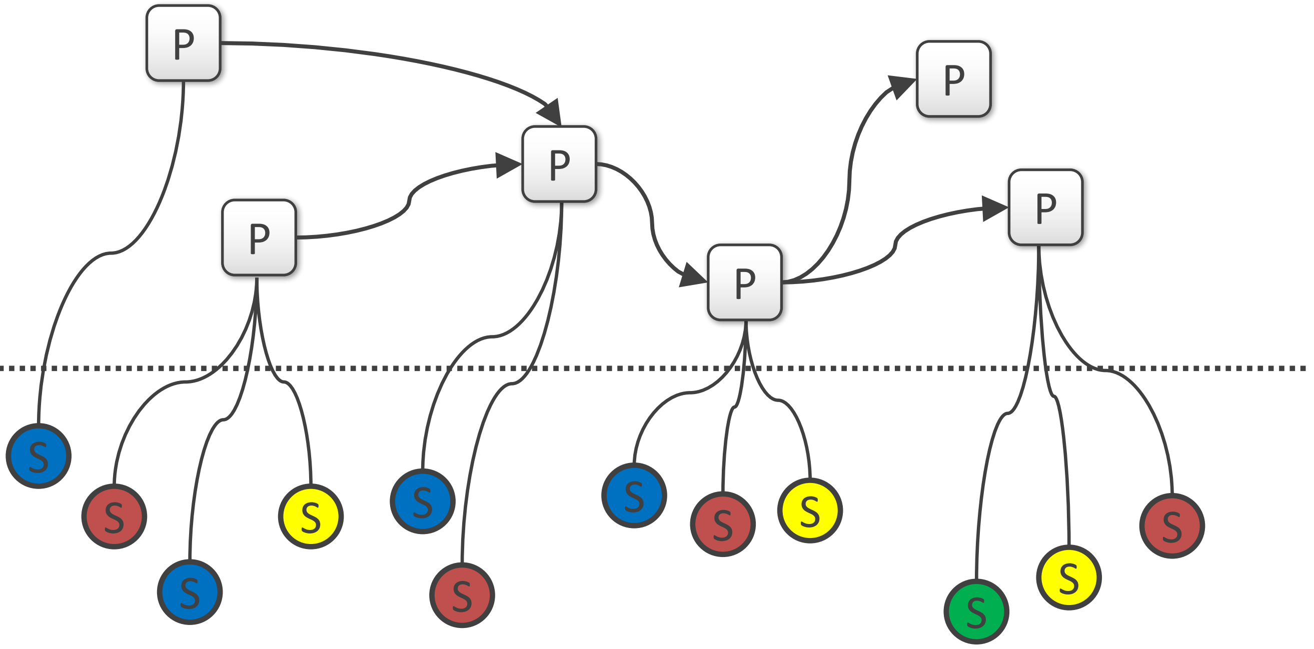
Types of Concrete Solutions

- Definition:

  The set of concrete solutions Σ related to a pattern language can be divided into a indexed family of sets $(A_i)_{i \in I}$ with $\forall A_i: A_i \subseteq \Sigma$. Thereby, each A_i represents a set of concrete solutions of a specific type. Concrete solutions of **different types differ in their essential characteristics**.

Pattern Language

Space of
Concrete Solutions
of a Pattern Language



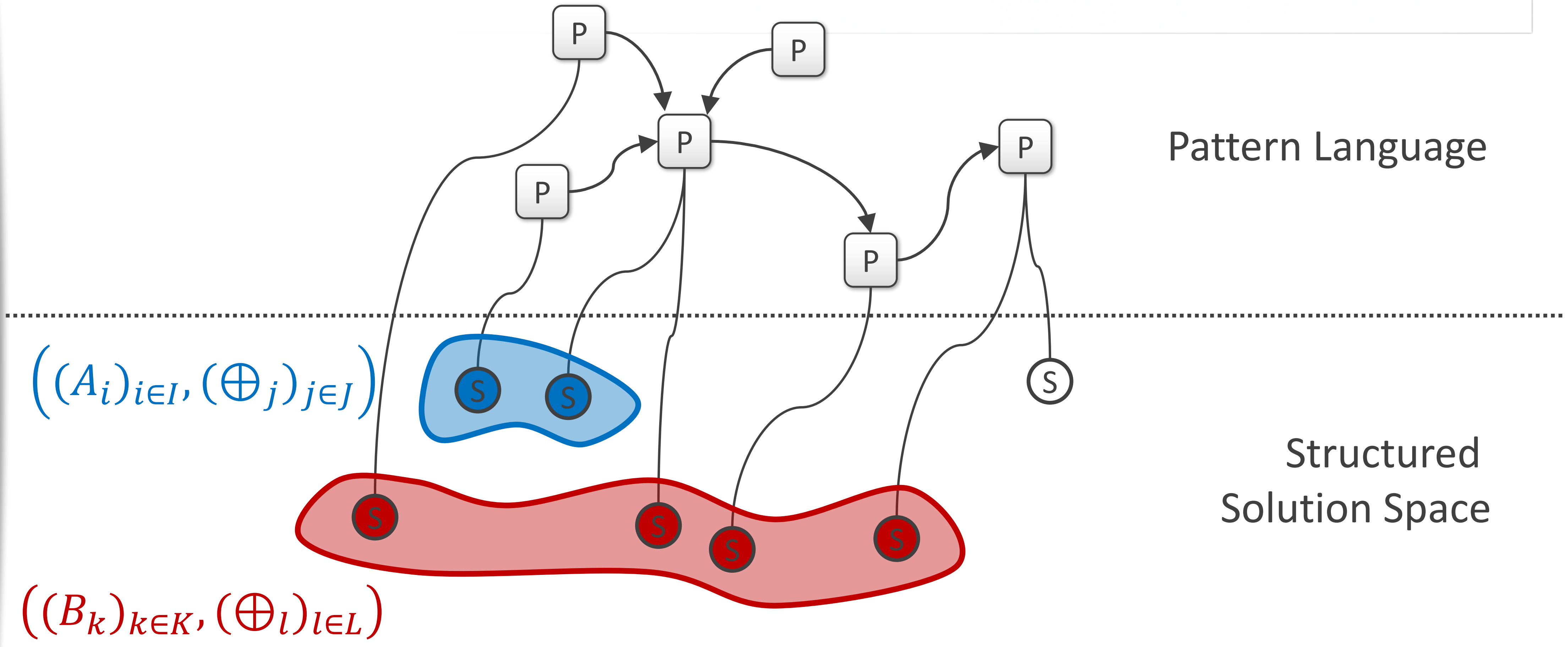
Solution Algebras

- Definition:

  A **solution algebra** is an ordered pair $\left((A_i)_{i\in I}, (\bigoplus_j)_{j\in J}\right)$ consisting of an indexed **family of concrete solutions** (carrier sets) $(A_i)_{i\in I}$ and an indexed **family of finitary aggregation operators** $(\oplus_j)_{j\in J}$.

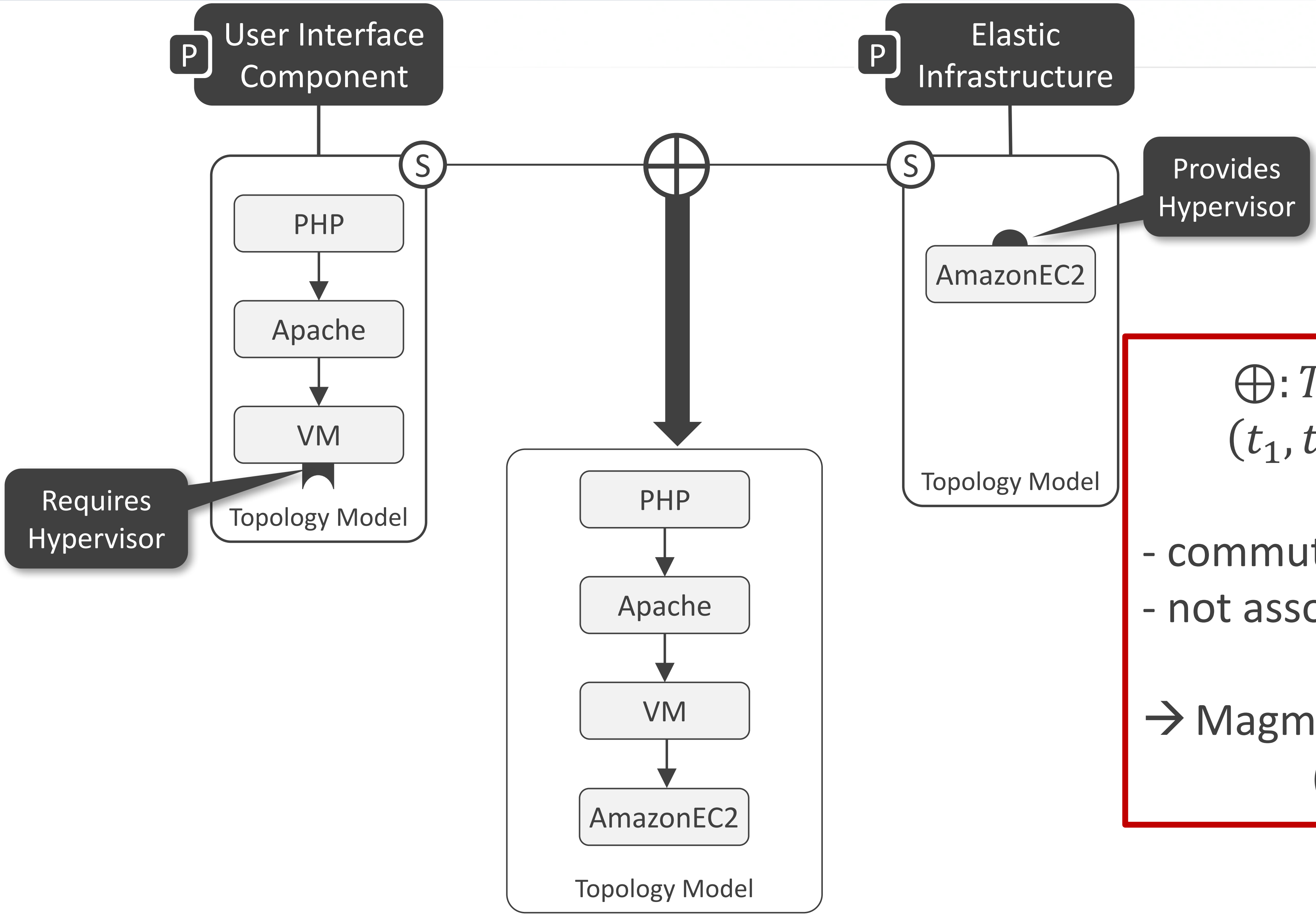
  For each $j \in J$, $\oplus_j$ is an operator $A_{i_1}\times\ldots\times A_{i_k} \rightarrow A_m$, $\oplus_j(a_1, \ldots, a_k) \mapsto a_m$.
  The $(n+1)$-tuple $(i_1, \ldots, i_n; k) \in I^{n+1}$associated with each operator $\oplus_j$ is referred to as the **type of the operator**. Operators combine elements of the cartesian product of the carrier sets and map them to an element of one of the carrier sets.

Solution Algebras


P
P
P
P
P
P
Pattern Language
$\left((A_i)_{i \in I}, (\bigoplus_j)_{j \in J}\right)$
S
S
S
Structured Solution Space
S
S
S
S
$\left((B_k)_{k \in K}, (\bigoplus_l)_{l \in L}\right)$

Some Aggregation Operators from the Domains of Cloud Application Provisioning, Cloud Application Management, and Costumes in Films

Cloud Application Provisioning: Aggregation of Topology Models

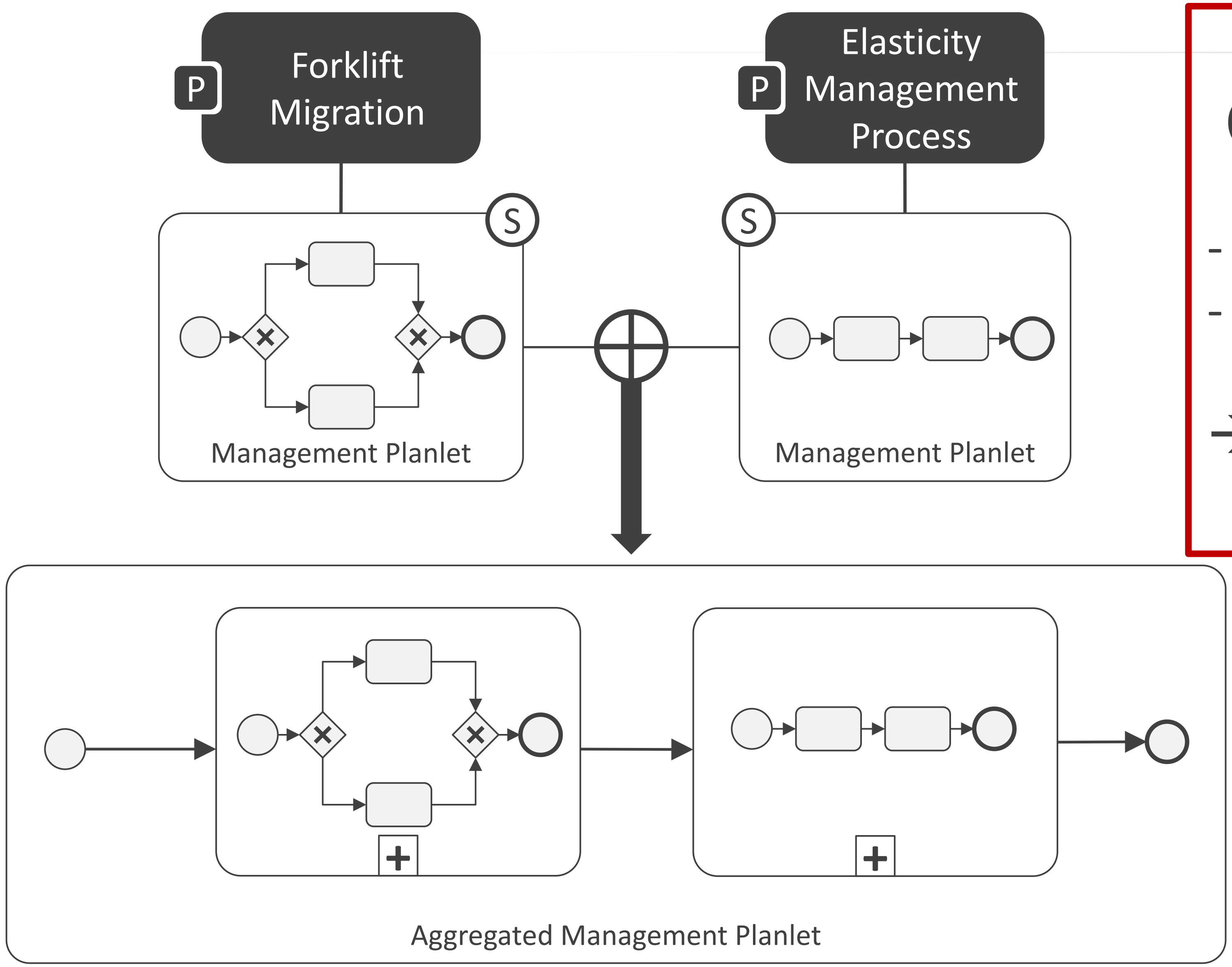


$$\oplus : T \times T \to T$$
$$(t_1, t_2) \mapsto t_m$$

- commutative
- not associative

→ Magma/Groupoid (T, ⊕)

Cloud Application Management: Aggregation of Management Planlets

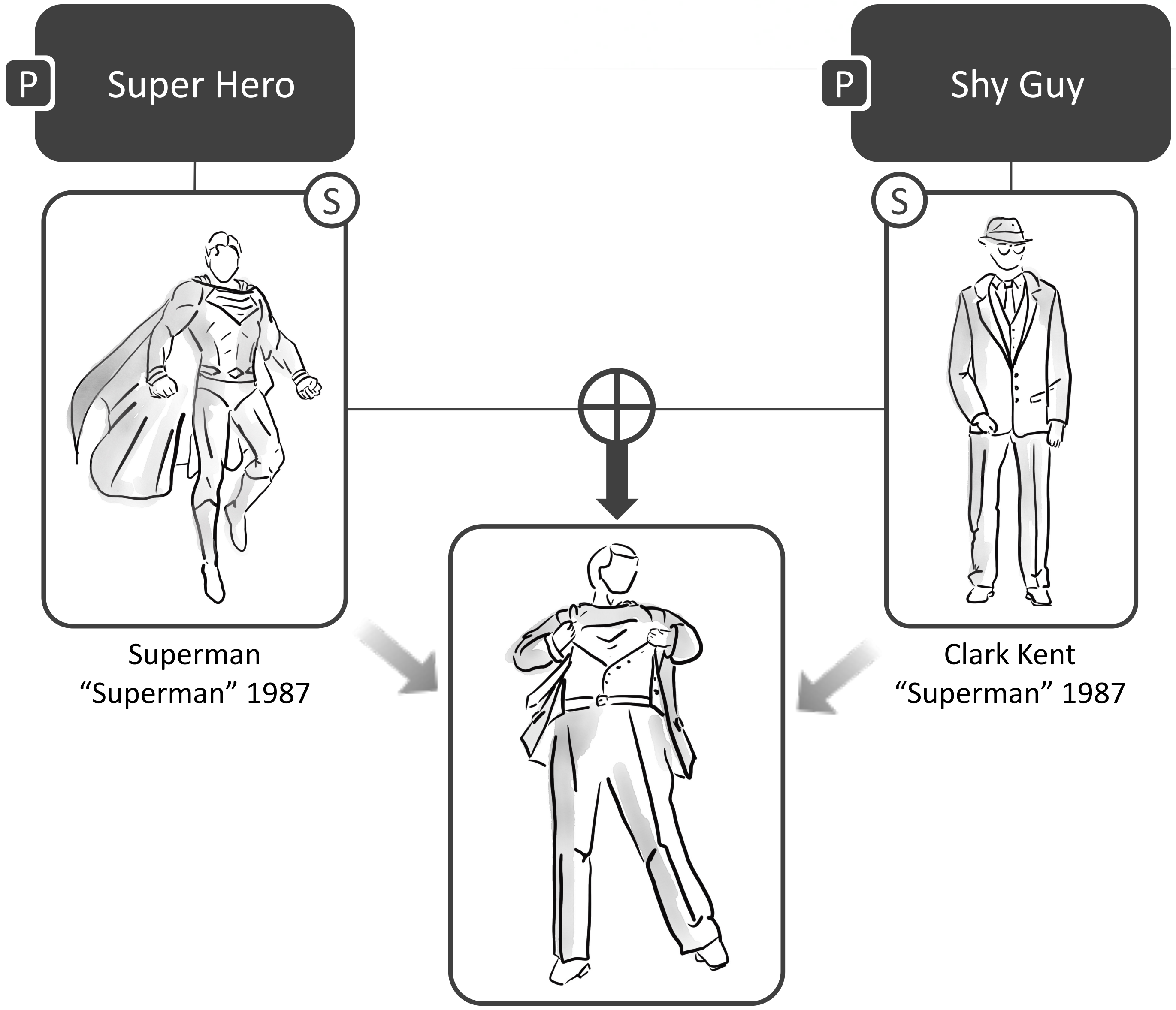


$$\oplus: MP \times MP \to MP$$
$$(mp_1, mp_2) \mapsto mp_m$$

- not commutative
- associative

→ Semigroup (MP, ⊕)

Costume Design: Aggregation of Costume Baseelements



$$\oplus_1 : C \times C \rightarrow C$$
$$(c_1, c_2) \mapsto c_1 \cup c_2$$

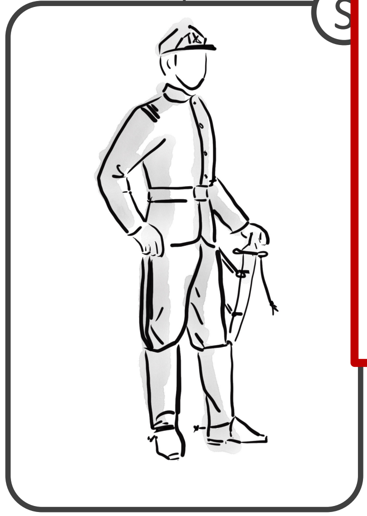
- commutative
- associative

Result: Double Identity: Clark Kent and Superman, “Superman” 1987

Costume Design: Distinct Aggregation of Costume Baseelements

P American Civil War Lieutenant

P Lakota Indian



1st Lt. John Dunbar "Dances with Wolves"1990

Lakota Man "Dances with Wolves"1990

Solution Algebra for Costumes

$$(C, \{\bigoplus_1, \bigoplus_2\})$$

$\bigoplus_1$ and $\bigoplus_2$ not distributive

$$\bigoplus_2 : C \times C \to C$$
$$(c_1, c_2) \mapsto c_1 \cup c_2$$

$$b_j, b_k \in c_1 \cup c_2 :$$
$$(b_j) \neq \omega(b_k)$$
$$(b_j \in c_1 \wedge b_k \in c_2)$$
$$\vee (b_j \in c_2 \wedge b_k \in c_1)$$

$$\omega : B \to T$$

- commutative
- associative

Result: Semi-Indian: John Dunbar as Lakota Man, "Dances with Wolves"1990

Conclusion and Future Work

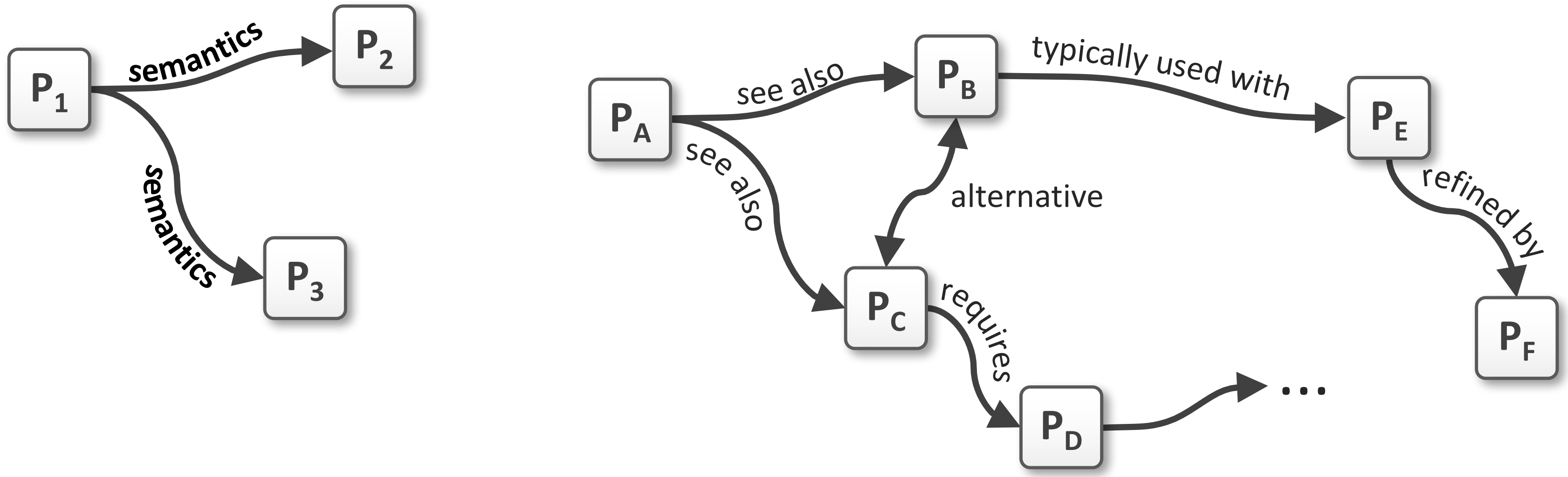
- Conclusion
  - Aggregation operators and solution algebras are the reflection of pattern languages on the level of concrete solutions
  - Solution Algebras can help to structure and organize the solution space of a pattern language
  - Automation of pattern language application can be based on solution algebras
- Future Work
  - We are going to implement an integrated pattern and solution repository
  - Repository will allow to aggregate concrete solutions
  - We are going to investigate how aggregation operators can help to conclude the quality, i.e., applicability of a pattern language

I am looking forward for discussions at the poster session?

IAAS Research

Background: Pattern Languages

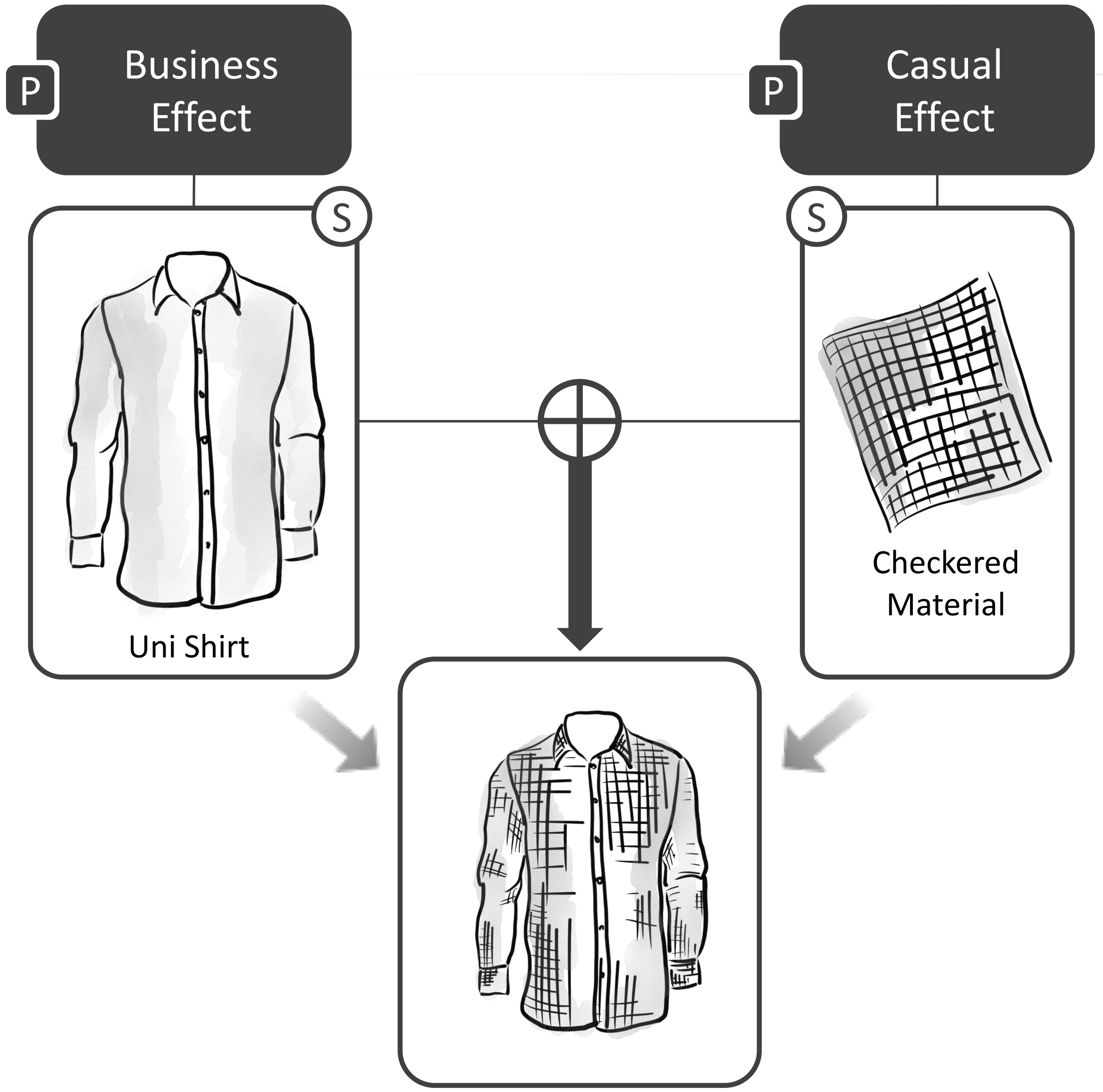
- Patterns are organized into Pattern Languages to **solve problems in combination**



- Pattern Languages provide sophisticated means to
  - Navigate purposefully between patterns
  - Ease their combined application

Zdun, “Systematic pattern selection using pattern language grammars and design space analysis,” Software: Practice and Experience, vol. 37, 2007
Falkenthal et al., “Leveraging Pattern Applications via Pattern Refinement,” Pursuit of Pattern Languages for Societal Change (PURPLSOC), 2015

Costume Design: Aggregation of Effects



Result: Casual Clothing: Checkered Shirt