Image Interpretation as Configuration

Image Interpretation as a Configuration Problem

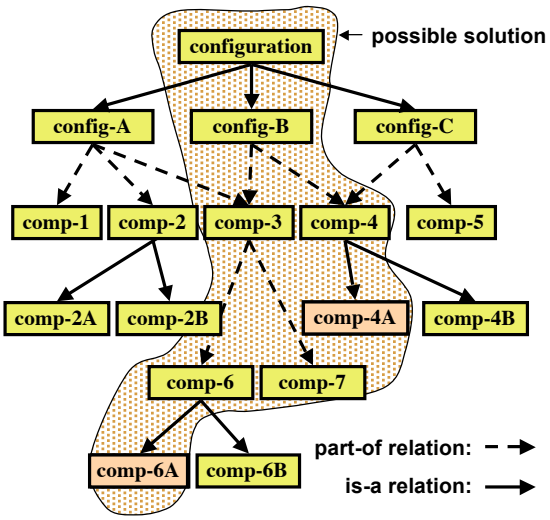
What is a configuration problem?

Construct an aggregate (a configuration) given
- **generic descriptions of parts**
- **compatibility constraints between parts**
- **a concrete task description**

Image interpretation may be viewed as constructing a "scene aggregate" which
- **meets generic constraints and**
- **incorporates parts prescribed by the concrete task**

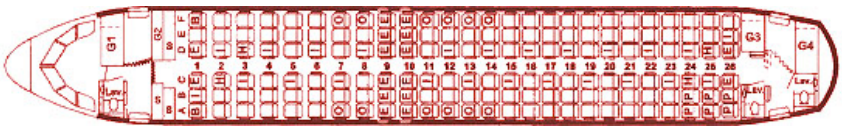
Methods and tools of configuration technology may be exploited

Illustration of Configuration



- boxes (frames) specify aggregate and component properties
- has-part relations bind components to aggregates
- is-a relations describe variants of entities
- constraints between entities (not shown) restrict choices and parameter combinations

A Real Configuration Task



Placement of cabin equipment (seats, kitchens, toilets, etc.) in view of

- customer wishes
- technical constraints
- legal constraints
- optimality criteria

Example of Concept Definition in KONWERK

KONWERK is a configuration system prototype developed at the AI Lab (LKI) of Hamburg University in 1986 - 1994. The commercial system Engcon has been developed based on KONWERK.

Concept "galley" describes service station in Airbus A340

```
def-concept
   :name galley
   :super-concept {cabin-interior-component rectangle}
   :parameters
       ref-nr [integer 2531000 2533999]
       door {1 2 4}
       trolleys {0 2 3 4 5 6 7 8 9 10}
       half-size-trolleys {0 1 2 3 4 5}
       meals [integer 28 140]
        type {longitudinal transversal}
       height {full half} (default 'full)
   :relations
       part-of [passenger-class]
```

Example: Concept Definition

```
(ist!  (eine Klasse)
       (ein Konstruktionsobjekt
       (Teil-von     (eine Passagierkabine)
       (Hat-Teile    #{#[(ein Einrichtungsgegenstand) 0 433] ':=
                     #[(eine Küche) 0 10]
                     #[(eine Toilette) 1 12]
                     #[(ein Flugbegleitersitz 1 16]
                     #[(ein Passagiersitz 5 395]})
       (Sitzabstand        [28inch 62inch])
       (Sitze/Reihe        {5 6 7 8 9})
       (Passagiere/Toilette      [15 60])
       (Passagiere|Flugbegleiter       [8 50])
       (Mahlzeiten/Passagier     [0 6])))
```

Object Descriptors

Object descriptors define object classes (concepts) by specifying possible instances. (Compare with concept expressions in a DL).

Specific values:	red, 35t, car37
Choice sets:	{red yellow green black blue}
Intervals:	[10km/h 300km/h]
Predicates:	(:satifies evenp)
Concepts:	(a car) (a chassis (axle_load [10t 40t]))
Atomic concepts:	(a symbol (self {red yellow green black blue})) (a number (self [0 inf]))
Logical operators:	(:and [50 100] (:satisfies evenp))

Set Descriptors

Set: (:set (a motor) (a body) (a chassis))

short form for

```
(:set   #[(a car_part) 3 3]
        #[(a motor) 1 1]
        #[(a body) 1 1]
        #[(a chassis) 1 1])
```

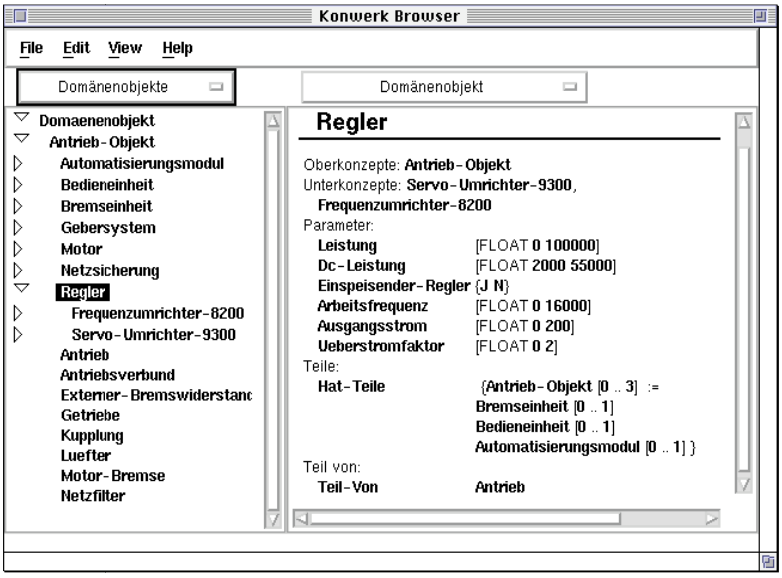
Subset: (:some (a motor) 2 4) ≈ #[(a motor) 2 4]

Decomposition Relations

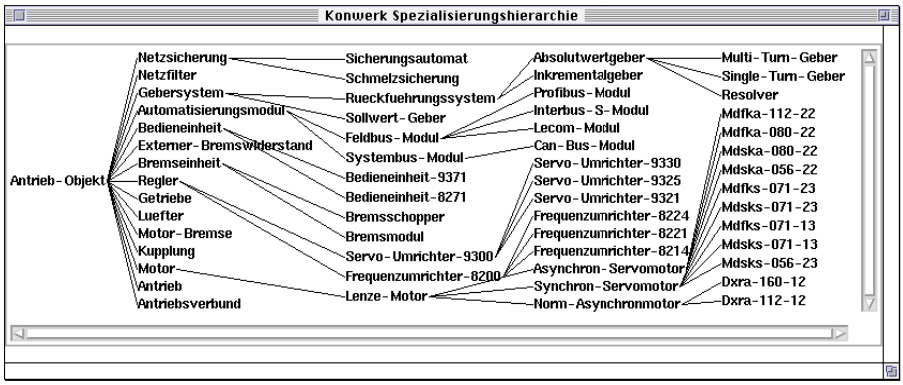
```
(is     (a car)
        (an object
                (has-parts   (:set  (a motor)
                                    (a body)
                                    (a chassis)))))
```

```
(is     (a car)
        (an object
                (has-motor          (a motor)
                (has-body           (a body)
                (has-chassis        (a chassis)))))
```

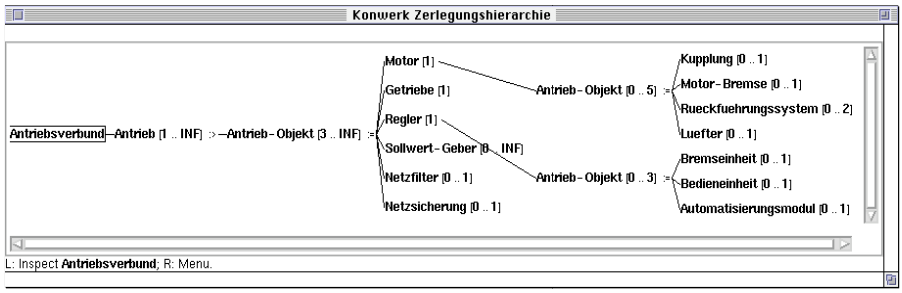
KONWERK Browser



KONWERK Specialization Hierarchy



KONWERK Decomposition Hierarchy



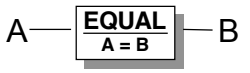
Representation Language of KONWERK

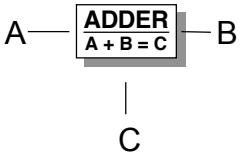
Language constructs can be mapped to logical constructs of a description logic by using:

- **Conjunction**
- **Negation and disjunction with atomic concepts**
- **Value restrictions**
- **Qualifying number restrictions**
- **Inverse roles**
- **Sets**
- **Concrete domains over R**

Constraint-based Configuration

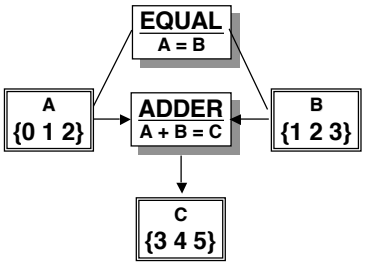
- Constraints represent **relations** between parameters or concepts
- Constraints are **multi-directional**
- Generating a constraint **network** (system of equations)
- **Consistency** check for value settings
- Restricting value ranges by **propagation**
- Computing all **solutions** by using constraint-satisfaction technologies
- **Incrementally increasing** constraint net

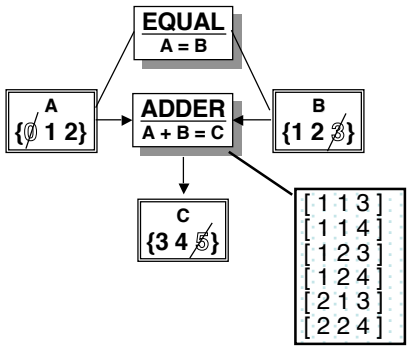




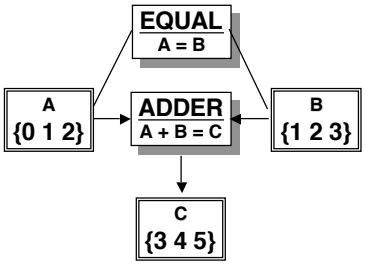
Constraint Propagation (1)

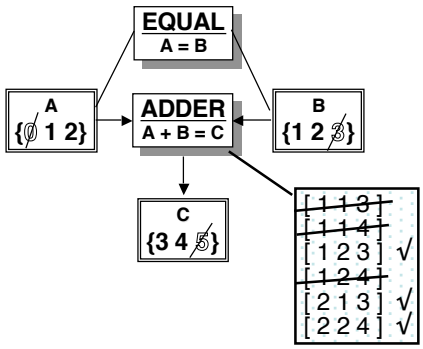
Contraint propagation ensures local consistency



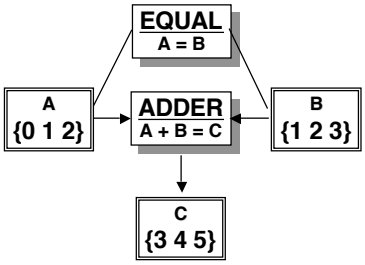


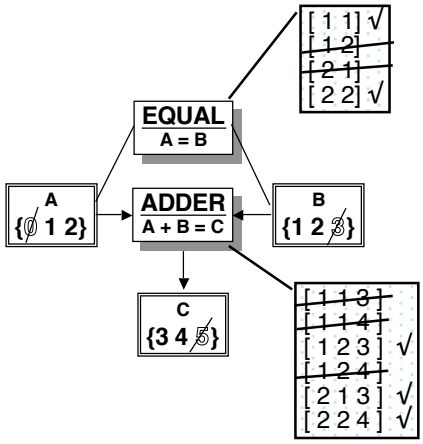
Constraint Propagation (2)



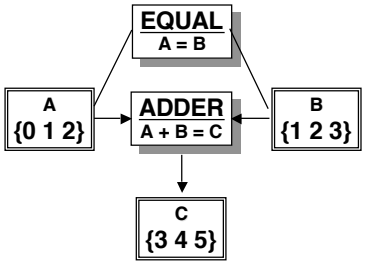


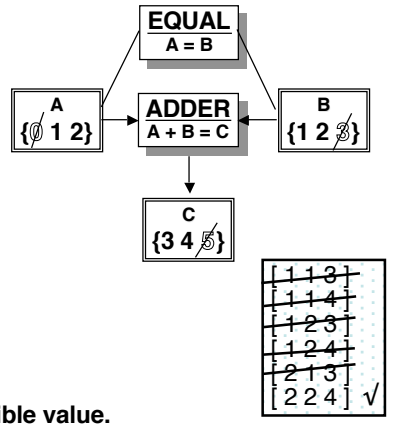
Constraint Propagation (3)





Constraint Propagation (4)





Finally, each variable has one possible value.

No constraint processing

Constraints in KONWERK

- **Constraint classes**
  Predominantlydomain-independent computational procedures
  Examples: adder, multiplier, sum, equal
- **Conceptual constraints**
  Description of a domain-specific constraint type, instantiation rules
  Example: motor displacement = sum of cylinder displacements
- **Constraint instances**
  Dynamically generated at configuration time
- **Constraint net**
  Propagates values through all constraint instances, recognizes conflicts

Examples of Conceptual Constraints

The displacement of a cylinder is computed as C = D^2 x π/4 x stroke height.

```
(constrain      ((#?C (a cylinder)))
                (square (#?C diameter) ?D)
                (multiply ?D π/4 ?P)
                (multiply ?P (#?C stroke)(#? displacement)))
```

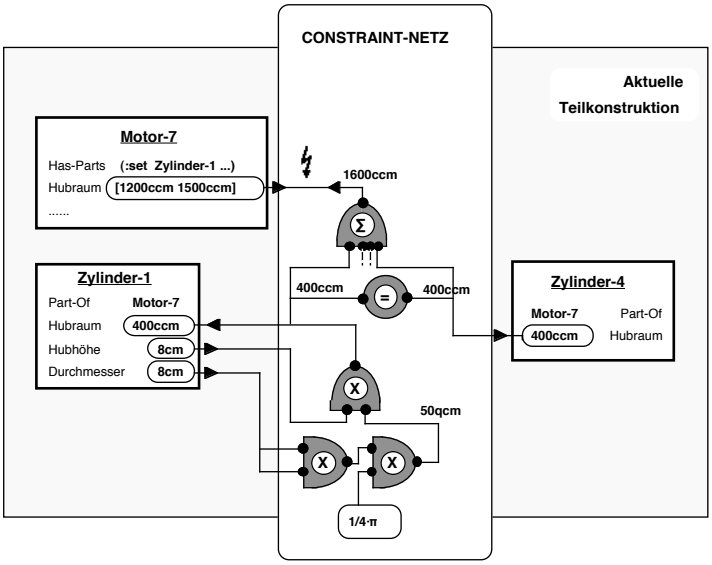
The displacement of a motor equals the sum of the displacements of its cylinders which are all equal.

```
(constrain      ((#?M (a motor)) (#?C :all (a cylinder (part-of #?M))))
                (all-equal (#?C displacement))
                (sum (#?M displacement) (#?C displacement)))
```

A car which is exported to Austria must have a catalytic converter.

```
(constrain      ((#?A (a car (export-to 'Austria))) (#?M (a motor (part-of #?A))))
                (exist (a cat (part-of #?M))))
```

Constraint Net

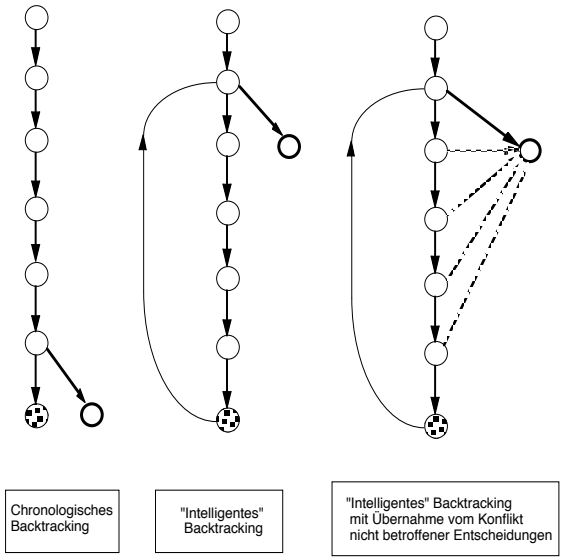


Central Configuration Cycle

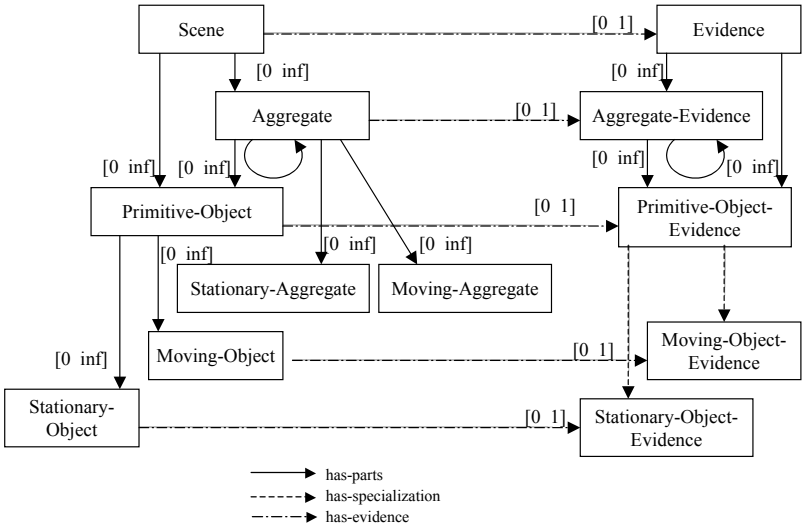
1. Select a partial configuration
2. Determine agenda of executable configuration steps
3. Select a configuration step
4. In case of choices, use one of the processes:
   - default value assignment
   - computational procedure
   - user interaction
   - library solution
   - local breadth-first search
5. Constraint propagation and conflict recognition

Backtracking

In case of a conflict, backtracking occurs. One may select one of 3 backtracking strategies:



Upper Model for Scene Interpretation



Concepts for Table-top Scenario

(only some concepts shown in figure)

- **`Scene` is instantiated for concrete scene interpretation**
- **Context knowledge (e.g. daytime of scene) constrains `Scene` instance**
- **Instances of `Primitive-Object-Evidence` are generated by low-level vision system**
- **Instances of scene concepts are instantiated by stepwise configuration process**