

On Mereologies in Computing Science
Syntax and Semantics
A Divertimento

Dines Bjørner
Fredsvej 11, DK-2840 Holte, Danmark
E-Mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~db

April 26, 2009: 16:04

Summary

In this talk we solve the following problems:

- we give a formal model of a large class of mereologies,
 - with simple entities modelled as parts
 - and their relations by connectors;
- we show that that class applies to a wide variety of societal infrastructure component domains;
- we show that there is a class of **CSP** channel and process structures that correspond to the class of mereologies where
 - mereology parts become **CSP** processes and
 - connectors become channels;
 - and where simple entity attributes become process states.
- We have yet to prove to what extent the models satisfy
 - the axiom systems for mereologies of, for example, (Casati&Varzi 1999)
 - and a calculus of individuals (Bowman&Clarke 1981).

1. Introduction

1.1. Physics and Societal Infrastructures

1.1.1. Physics

- Physics study that of nature which can be measured
 - within us,
 - around us and
 - between ‘within’ and ‘around’!
- To make mathematical models of physics phenomena,
 - physics has helped develop and uses mathematics,
 - notably calculus and statistics.

[**1. Introduction**, **1.1. Physics and Societal Infrastructures**]

1.1.2. Societal Infrastructures

- Domain engineering primarily studies societal infrastructure components which can be
 - reasoned about,
 - built and
 - manipulated by humans.
- To make domain models of infrastructure components, domain engineering makes use of
 - formal specification languages,
 - their reasoning systems: formal testing, model checking and verification, and
 - their tools.

[**1. Introduction**]**1.2. Structures****1.2.1. In Nature**

- Physics turns to algebra in order to handle structures in nature.
 - Algebra appears to be useful in a number of applications, to wit:
 - * the abstract modelling of chemical compounds.
 - But there seems to be many structures in nature
 - * that cannot be captured in a satisfactory way by mathematics, including algebra
 - * and when captured in discrete mathematical disciplines such as sets, graph theory and combinatorics
 - the “integration” of these mathematically represented — structures
 - with calculus (etc.) — becomes awkward;
 - well, I know of no successful attempts.

[**1. Introduction**, **1.2. Structures**]

1.2.2. In Society

- Domain engineering turns to discrete mathematics —
 - as embodied in formal specification languages
 - and as “implementable” in programming languages —in order to handle structures in societal infrastructure components.
- These languages allow
 - (a) the expression of arbitrarily complicated structures,
 - (b) the evaluation of properties over such structures,
 - (c) the “building & demolition” of such structures, and
 - (d) the reasoning over such structures.
- They also allow the expression of dynamically varying structures
 - something mathematics is “not so good at” !

[**1. Introduction**, **1.2. Structures**, **1.2.2. In Society**]

- But the specification languages have two problems:
 - (i) they do not easily, if at all,
 - * handle continuity, that is, they do not embody calculus,
 - * or, for example, statistical concepts, etc.,and
 - (ii) they handle
 - * actual structures of societal infrastructure components
 - * and attributes of atomic and composite entities of these –
 - usually by identical techniques
 - thereby blurring what we think is an important distinction.

[**1. Introduction**]**1.3. Structure of This Talk**

- The rest of the talk is organised as follows.
- First we give a first main, a meta-example,
 - of syntactic aspects of a class of mereologies.
- Then we discuss concepts of atomic and composite simple entities.
- We then “perform”
 - the ontological trick of mapping the assembly and unit entities
 - and their connections
 - exemplified in the first main meta-example
 - into CSP processes and channels, respectively —
 - the second and last main — meta-example
 - * of semantic aspects of a class of mereologies.

2. A Syntactic Model of a Class of Mereologies

2.1. Systems, Assemblies, Units

- We speak of systems as assemblies.
- From an assembly we can immediately observe a set of parts.
- Parts are either assemblies or units.
- We do not further define what assemblies and units are.

type

$$S = A, A, U, P = A \mid U$$

value

$$\text{obs_Ps}: A \rightarrow \text{P-set}$$

- Parts observed from an assembly are said to be immediately embedded in, i.e., **within**, that assembly.
- Two or more different parts of an assembly are said to be immediately **adjacent** to one another.

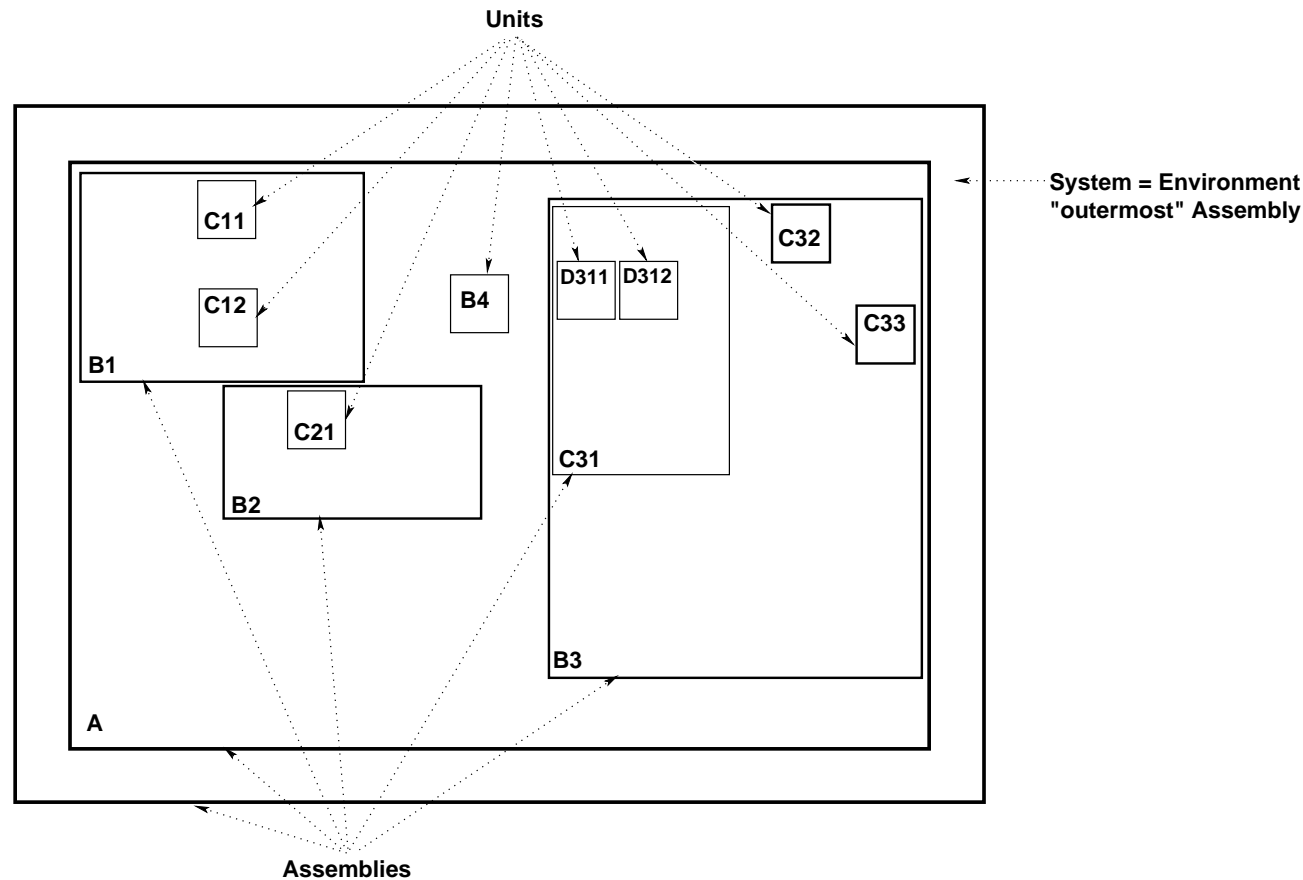
[**2. A Syntactic Model of a Class of Mereologies**, **2.1. Systems, Assemblies, Units**]

Figure 1: Assemblies and Units “embedded” in an Environment

- A system includes its environment.
- And we do not worry, so far, about the semiotics of all this !

[2. A Syntactic Model of a Class of Mereologies, 2.1. Systems, Assemblies, Units]

- Given obs_Ps we can define a function, xtr_Ps ,
 - which applies to an assembly \mathbf{a} and
 - which extracts all parts embedded in \mathbf{a} and including \mathbf{a} .
- The functions obs_Ps and xtr_Ps define the meaning of embeddedness.

value

$$\text{xtr_Ps}: A \rightarrow \text{P-set}$$

$$\text{xtr_Ps}(\mathbf{a}) \equiv$$

$$\mathbf{let} \text{ ps} = \{\mathbf{a}\} \cup \text{obs_Ps}(\mathbf{a}) \mathbf{in} \text{ ps} \cup \mathbf{union}\{\text{xtr_Ps}(\mathbf{a}') \mid \mathbf{a}':A \cdot \mathbf{a}' \in \text{ps}\} \mathbf{end}$$

- **union** is the distributed union operator.

[**2. A Syntactic Model of a Class of Mereologies**, **2.1. Systems, Assemblies, Units**]

- Parts have unique identifiers.
- All parts observable from a system are distinct.

type

AUI

value

obs_AUI: $P \rightarrow \text{AUI}$

axiom

$\forall a:A \cdot$

let ps = obs_Ps(a) **in**

$\forall p',p'':P \cdot \{p',p''\} \subseteq \text{ps} \wedge p' \neq p'' \Rightarrow \text{obs_AUI}(p') \neq \text{obs_AUI}(p'') \wedge$

$\forall a',a'':A \cdot \{a',a''\} \subseteq \text{ps} \wedge a' \neq a'' \Rightarrow \text{xtr_Ps}(a') \cap \text{xtr_Ps}(a'') = \{\} \quad \mathbf{end}$

[2. A Syntactic Model of a Class of Mereologies]

2.2. 'Adjacency' and 'Within' Relations

2.2.1. Immediate 'Adjacency'

- Two parts, p, p' , are said to be *immediately next to*, i.e., $i_next_to(p, p')(a)$, one another in an assembly a
 - if there exists an assembly, a' equal to or embedded in a
 - such that p and p' are observable in that assembly a' .

value

$$i_next_to: P \times P \rightarrow A \xrightarrow{\sim} \mathbf{Bool}$$

$$i_next_to(p, p')(a) \equiv$$

$$\exists a': A \cdot a' = a \vee a' \in xtr_Ps(a) \cdot \{p, p'\} \subseteq obs_Ps(a')$$

pre $p \neq p'$

[**2. A Syntactic Model of a Class of Mereologies**, **2.2. 'Adjacency' and 'Within' Relations**]

2.2.2. Immediate 'Within'

- One part, p , is said to be *immediately within* another part, p' , i.e., $i_within(p,p')(a)$, in an assembly a
 - if there exists an assembly, a' equal to or embedded in a
 - such that p is observable in a' .

value

$i_within: P \times P \rightarrow A \xrightarrow{\sim} \mathbf{Bool}$

$i_within(p,p')(a) \equiv$

$\exists a':A \cdot (a=a' \vee a' \in xtr_Ps(a)) \cdot p'=a' \wedge p \in obs_Ps(a')$

[2. A Syntactic Model of a Class of Mereologies, 2.2. 'Adjacency' and 'Within' Relations]

2.2.3. Transitive 'Within'

- We can generalise the immediate 'within' property.
- A part, p , is (transitively) within a part p' , $\text{within}(p,p')(a)$, of an assembly, a ,
 - either if p , is immediately within p' of that assembly, a ,
 - or if there exists a (proper) part p'' of p'
 - such that $\text{within}(p'',p)(a)$.

value

$\text{within}: P \times P \rightarrow A \xrightarrow{\sim} \mathbf{Bool}$

$\text{within}(p,p')(a) \equiv$

$i_within(p,p')(a) \vee \exists p'':P \cdot p'' \in \text{obs_Ps}(p) \wedge \text{within}(p'',p')(a)$

[**2. A Syntactic Model of a Class of Mereologies**, **2.2. 'Adjacency' and 'Within' Relations**, **2.2.3. Transitive 'Within'**]

- The function **within** can be defined, alternatively,
- using **xtr_Ps** and **i_within**
- instead of **obs_Ps** and **within** :

value

within': $P \times P \rightarrow A \xrightarrow{\sim} \mathbf{Bool}$

within'(p,p')(a) \equiv

i_within(p,p')(a) $\vee \exists p'' : P \cdot p'' \in \mathbf{xtr_Ps}(p) \wedge \mathbf{i_within}(p'',p')(a)$

lemma: **within** \equiv **within'**

[2. A Syntactic Model of a Class of Mereologies, 2.2. 'Adjacency' and 'Within' Relations]

2.2.4. Transitive 'Adjacency'

- We can generalise the immediate 'next to' property.
- Two parts, p , p' of an assembly, a , are adjacent if they are
 - either 'next to' one another
 - or if there are two parts p_o , p'_o
 - * such that p , p' are embedded in respectively p_o and p'_o
 - * and such that p_o , p'_o are immediately next to one another.

value

adjacent: $P \times P \rightarrow A \xrightarrow{\sim} \mathbf{Bool}$

adjacent(p, p')(a) \equiv

i_next_to(p, p')(a) \vee

$\exists p'', p''': P \cdot \{p'', p'''\} \subseteq_{\text{xtr}} Ps(a) \wedge \text{i_next_to}(p'', p''')(a) \wedge$

$((p=p'') \vee \text{within}(p, p'')(a)) \wedge ((p'=p''') \vee \text{within}(p', p''')(a))$

[**2. A Syntactic Model of a Class of Mereologies**]**2.3. Mereology, Part I**

- So far we have built a *ground mereology* model, $\mathcal{M}_{\mathcal{G}\text{ground}}$.
- Let \sqsubseteq denote *parthood*, *x is part of y*, $x \sqsubseteq y$.

$$\forall x(x \sqsubseteq x)^1 \quad (1)$$

$$\forall x, y(x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow (x = y) \quad (2)$$

$$\forall x, y, z(x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow (x \sqsubseteq z) \quad (3)$$

- Let \sqsubset denote *proper parthood*, *x is part of y*, $x \sqsubset y$.
- Formula 4 defines $x \sqsubset y$. Equivalence 5 can be proven to hold.

$$\forall x \sqsubset y =_{\text{def}} x(x \sqsubseteq y) \wedge \neg(x = y) \quad (4)$$

$$\forall \forall x, y(x \sqsubseteq y) \Leftrightarrow (x \sqsubset y) \vee (x = y) \quad (5)$$

¹Our notation now is not RSL but some conventional first-order predicate logic notation.

[2. A Syntactic Model of a Class of Mereologies, 2.3. Mereology, Part I]

- The *proper part* ($x \sqsubset y$) relation is a strict partial ordering:

$$\forall x \neg(x \sqsubset x) \quad (6)$$

$$\forall x, y (x \sqsubset y) \Rightarrow \neg(y \sqsubset x) \quad (7)$$

$$\forall x, y, z (x \sqsubset y) \wedge (y \sqsubset z) \Rightarrow (x \sqsubset z) \quad (8)$$

- *Overlap*, \bullet , is also a relation of parts:

– Two individuals overlap if they have parts in common:

$$x \bullet y =_{\text{def}} \exists z (z \sqsubset x) \wedge (z \sqsubset y) \quad (9)$$

$$\forall x (x \bullet x) \quad (10)$$

$$\forall x, y (x \bullet y) \Rightarrow (y \bullet x) \quad (11)$$

[**2. A Syntactic Model of a Class of Mereologies**, **2.3. Mereology, Part I**]

- Proper overlap, \circ , can be defined:

$$x \circ y =_{\text{def}} (x \bullet x) \wedge \neg(x \sqsubseteq y) \wedge \neg(y \sqsubseteq x) \quad (12)$$

- Whereas Formulas (1-11) holds of the model of mereology we have shown so far, Formula (12) does not.
- In the next section we shall repair that situation.
- The *proper part* relation, \sqsubset , reflects the *within* relation.
- The *disjoint* relation, \oint , reflects the *adjacency* relation.

$$x \oint y =_{\text{def}} \neg(x \bullet y) \quad (13)$$

[2. A Syntactic Model of a Class of Mereologies, 2.3. Mereology, Part I]

- Disjointness is symmetric:

$$\forall x, y (x \not\sqcap y) \Rightarrow (y \not\sqcap x) \quad (14)$$

- The *weak supplementation* relation, Formula 15, expresses
 - that if y is a proper part of x
 - then there exists a part z
 - such that z is a proper part of x
 - and z and y are disjoint
- That is, whenever an individual has one proper part then it has more than one.

$$\forall x, y (y \sqsubset x) \Rightarrow \exists z (z \sqsubset x) \wedge (z \not\sqcap y) \quad (15)$$

[**2. A Syntactic Model of a Class of Mereologies**, **2.3. Mereology, Part I**]

- Formulas 1–3 and 15 together determine the *minimal mereology*, $\mathcal{M}_{\text{Minimal}}$.
- Formula 15 does not hold of the model of mereology we have shown so far.
- We shall comment on this once we have introduced the notion of parts having attributes.

[**2. A Syntactic Model of a Class of Mereologies**]

2.4. Connectors

- So far we have only covered notions of
 - parts being next to other parts or
 - within one another.
- We shall now add to this a rather general notion of parts being otherwise related.
- That notion is one of connectors.

[**2. A Syntactic Model of a Class of Mereologies**, **2.4. Connectors**]

- Connectors provide for connections between parts.
- A connector is an ability to be connected.
- A connection is the actual fulfillment of that ability.
- Connections are relations between pairs of parts.
- Connections “cut across” the “classical”
 - *parts being part of the (or a) whole* and
 - *parts being related by embeddedness or adjacency.*

[2. A Syntactic Model of a Class of Mereologies, 2.4. Connectors]

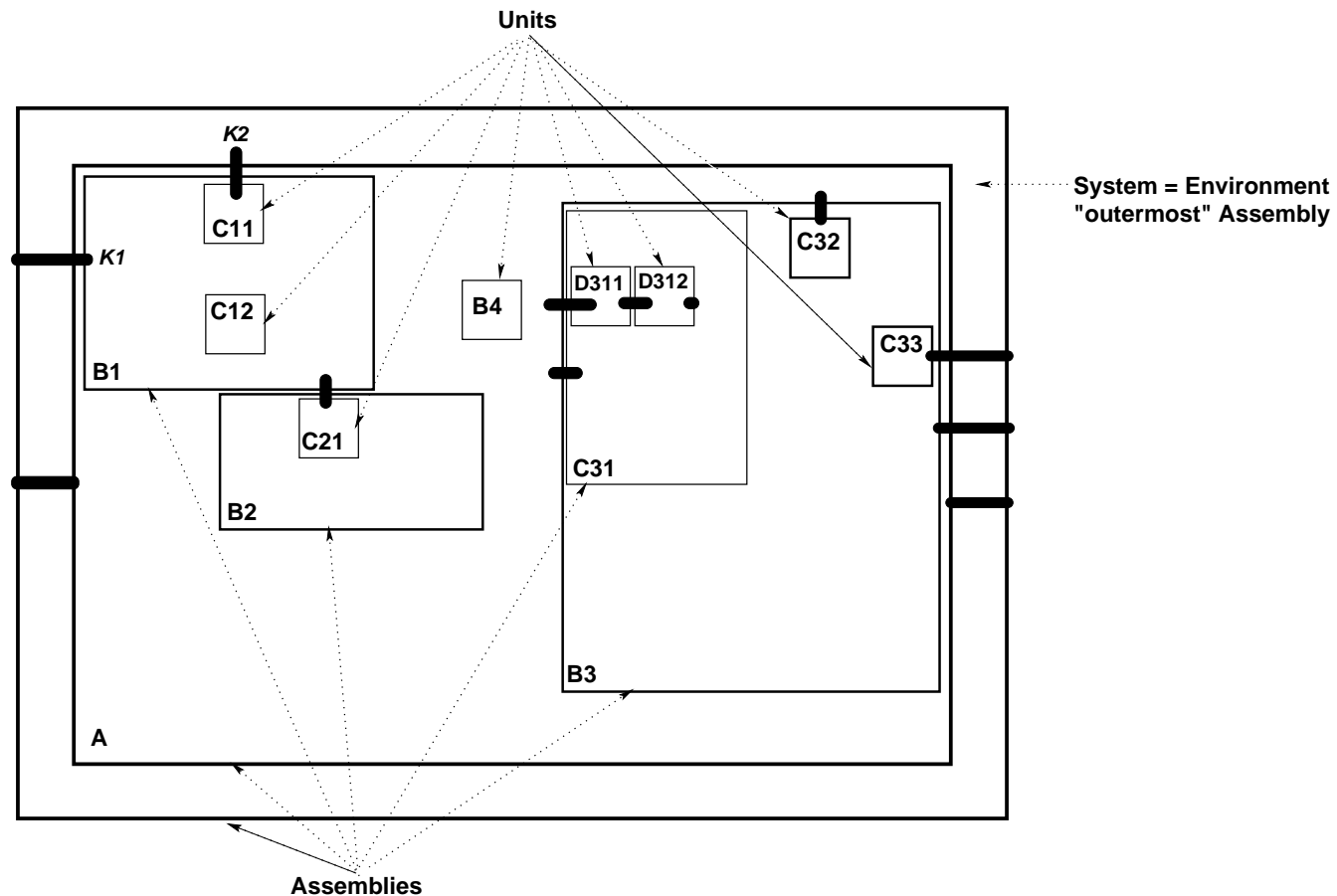


Figure 2: Assembly and Unit Connectors: Internal and External

- For now, we do not “ask” for the meaning of connectors !

[**2. A Syntactic Model of a Class of Mereologies**, **2.4. Connectors**]

- From a system we can observe all its connectors.
- From a connector we can observe
 - its unique connector identifier and
 - the set of part identifiers of the parts that the connector connects.
- All part identifiers of system connectors identify parts of the system.
- All observable connector identifiers of parts identify connectors of the system.

[**2. A Syntactic Model of a Class of Mereologies**, **2.4. Connectors**]**type**

K

value $\text{obs_Ks}: S \rightarrow \mathbf{K\text{-set}}$ $\text{obs_KI}: K \rightarrow \text{KI}$ $\text{obs_Is}: K \rightarrow \mathbf{AUI\text{-set}}$ $\text{obs_KIs}: P \rightarrow \mathbf{KI\text{-set}}$ **axiom** $\forall k:K \cdot \mathbf{card} \text{ obs_Is}(k)=2,$ $\forall s:S, k:K \cdot k \in \text{obs_Ks}(s) \Rightarrow \exists p:P \cdot p \in \text{xtr_Ps}(s) \Rightarrow \text{obs_AUI}(p) \in \text{obs_Is}(k),$ $\forall s:S, p:P \cdot \forall ki:KI \cdot ki \in \text{obs_KIs}(p) \Rightarrow \exists! k:K \cdot k \in \text{obs_Ks}(s) \wedge ki = \text{obs_KI}(k)$

- This model allows for a rather “free-wheeling” notion of connectors
 - one that allows internal connectors to “cut across” embedded and adjacent parts;
 - and one that allows external connectors to “penetrate” from an outside to any embedded part.

[**2. A Syntactic Model of a Class of Mereologies**, **2.4. Connectors**]

- We need define an auxiliary function.
 - $\text{xtr}\forall\text{KIs}(p)$ applies to a system
 - and yields all its connector identifiers.

value

$\text{xtr}\forall\text{KIs}: S \rightarrow \text{KI-set}$

$\text{xtr}\forall\text{Ks}(s) \equiv \{\text{obs_KI}(k) \mid k:K \cdot k \in \text{obs_Ks}(s)\}$

[2. A Syntactic Model of a Class of Mereologies]

2.5. Mereology, Part II

We shall interpret connections as follows:

- A connection between parts p_i and p_j
 - that enjoy a p_i **adjacent to** p_j relationship, means $p_i \circ p_j$,
 - that is, although parts p_i and p_j are **adjacent**
 - they do *share* “something”, i.e., have something *in common*.
 - What that “something” is we shall comment on later, when we have “mapped” systems onto parallel compositions of **CSP** processes.
- A connection between parts p_i and p_j
 - that enjoy a p_i **within** p_j relationship,
 - does not add other meaning than
 - commented upon later, again when we have “mapped” systems onto parallel compositions of **CSP** processes.

[**2. A Syntactic Model of a Class of Mereologies**, **2.5. Mereology, Part II**]

- With the above interpretation we may arrive at the following, perhaps somewhat “awkward-looking” case:
 - a connection connects two adjacent parts p_i and p_j
 - * where part p_i is within part p_{i_o}
 - * and part p_j is within part p_{j_o}
 - * where parts p_{i_o} and p_{j_o} are adjacent
 - * but not otherwise connected.
 - How are we to explain that !
 - * Since we have not otherwise interpreted the meaning of parts,
 - * we can just postulate that “so it is” !
 - * We shall , later, again when we have “mapped” systems onto parallel compositions of **CSP** processes, give a more satisfactory explanation.

3. Discussion & Interpretation

- Before a semantic treatment of the concept of mereology
 - let us review what we have done; and
 - let us interpret our abstraction
 - * (i.e., relate it to actual societal infrastructure components).

[**3. Discussion & Interpretation**]**3.1. What We have Done So Far ?**

- We have
 - presented a model that is claimed to abstract essential mereological properties of
 - * machine assemblies,
 - * railway nets,
 - * the oil industry,
 - * oil pipelines,
 - * buildings with installations,
 - * hospitals,
 - * etcetera.

[**3. Discussion & Interpretation**]

3.2. Six Interpretations

- Let us substantiate the claims made in the previous paragraph.
 - We will do so, albeit informally, in the next many paragraphs.
 - Our substantiation is a form of diagrammatic reasoning.
 - Subsets of diagrams will be claimed to represent parts, while
 - Other subsets will be claimed to represent connectors.
- The reasoning is incomplete.

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.1. Air Traffic

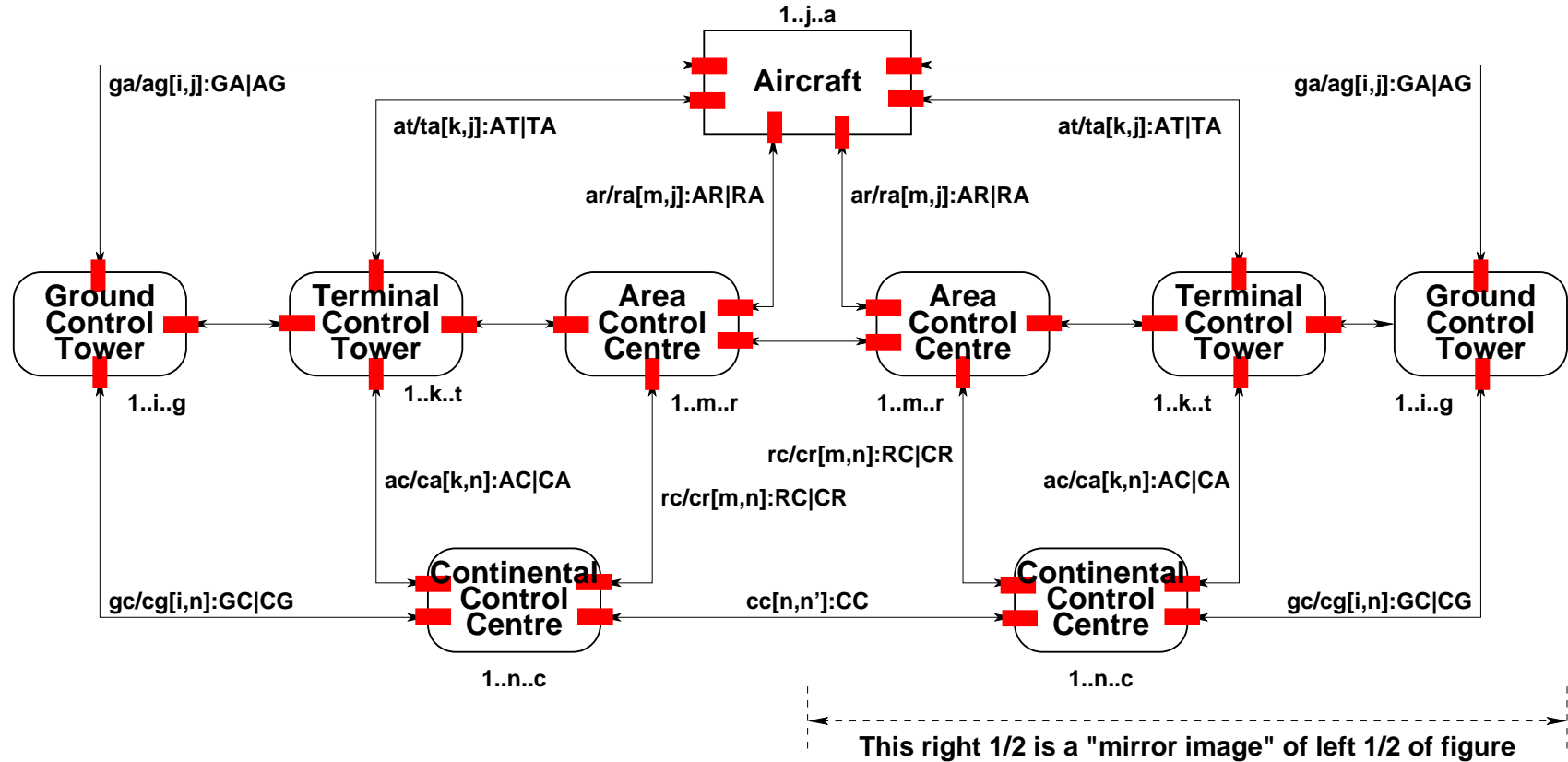


Figure 3: An air traffic system. Black boxes and lines are units; red boxes are connections

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.2. Buildings

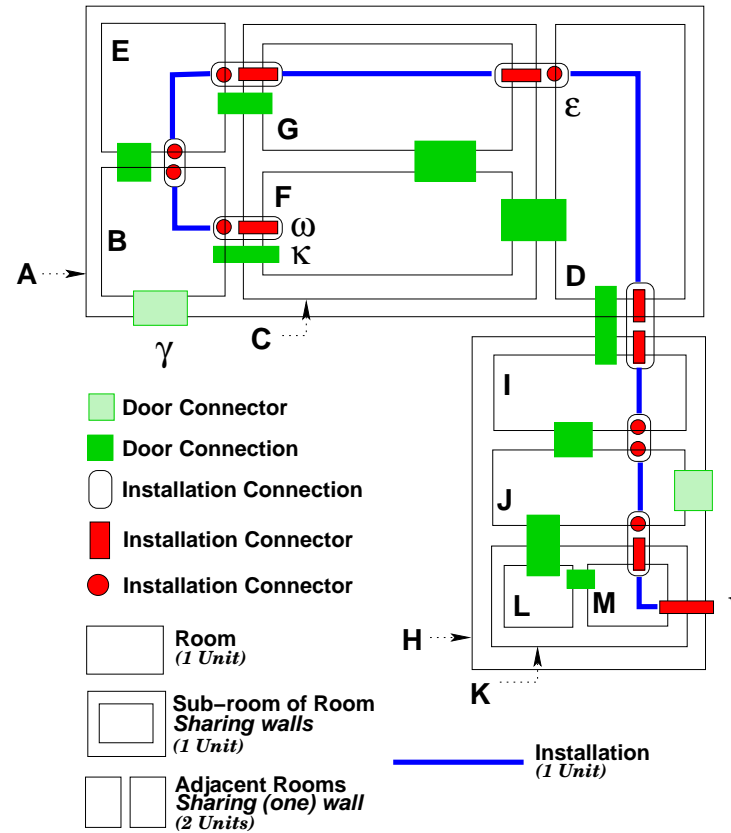


Figure 4: A building plan with installation

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.3. Financial Service Industry

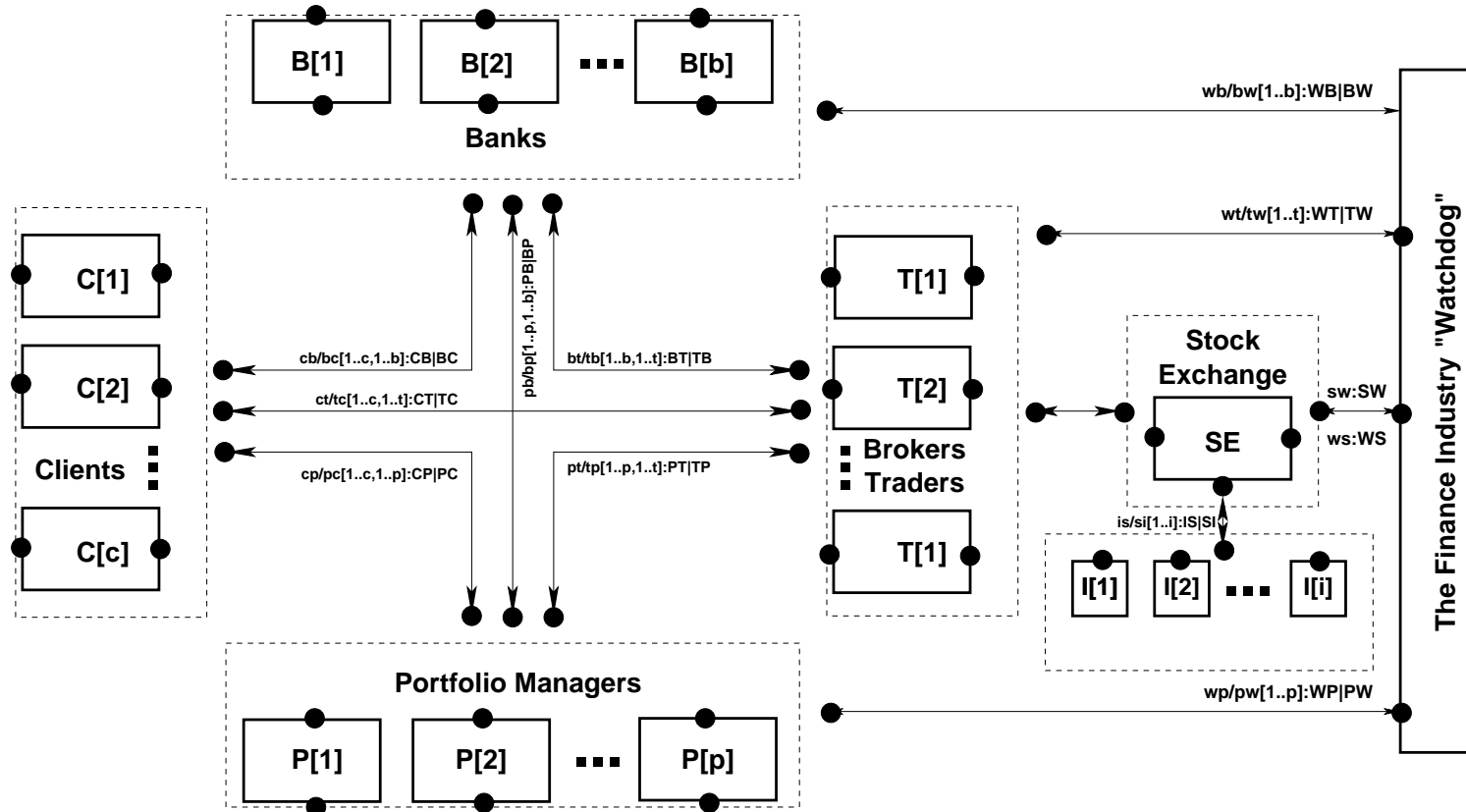


Figure 5: A financial service industry

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.4. Machine Assemblies

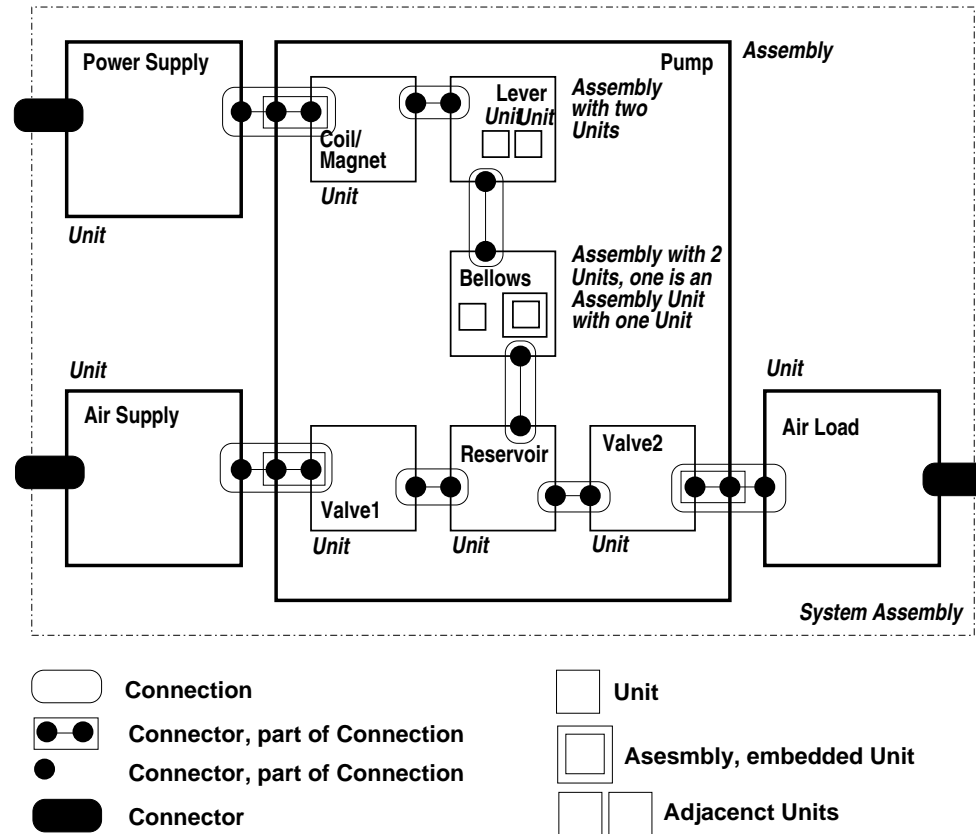


Figure 6: An air pump, i.e., a physical mechanical system

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.5. Oil Industry

“The” Overall Assembly

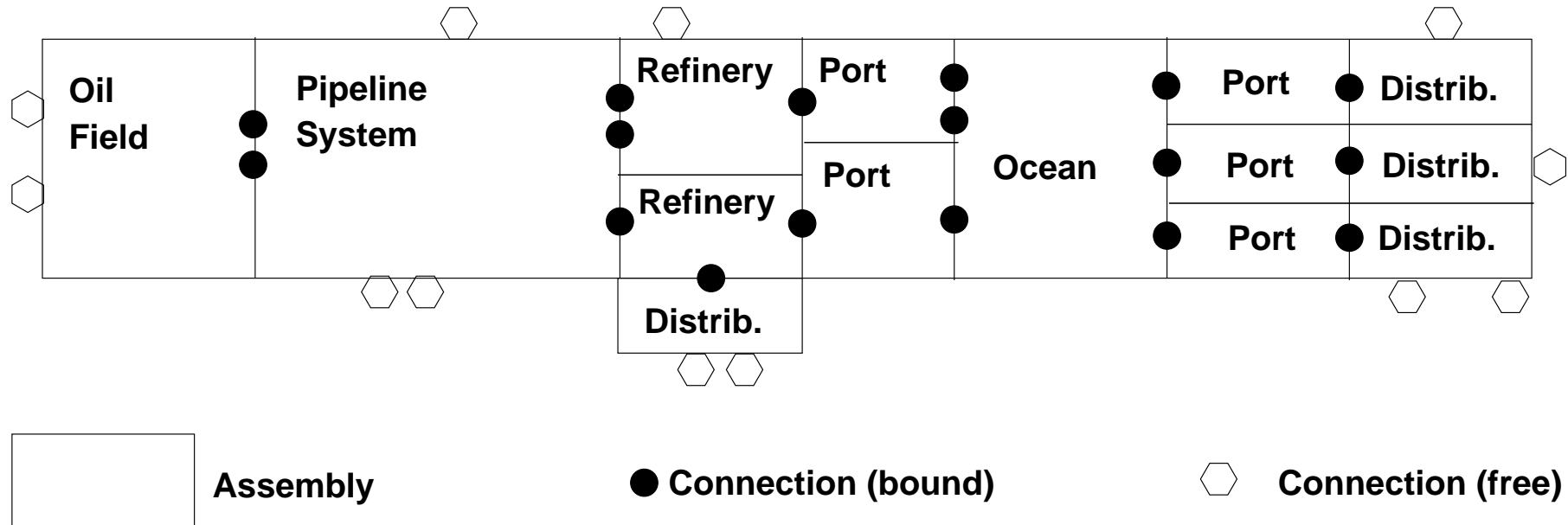


Figure 7: A Schematic of an Oil Industry

[3. Discussion & Interpretation, 2. Six Interpretations, 5. Interpretation, Oil Industry]

A Concretised Assembly Unit

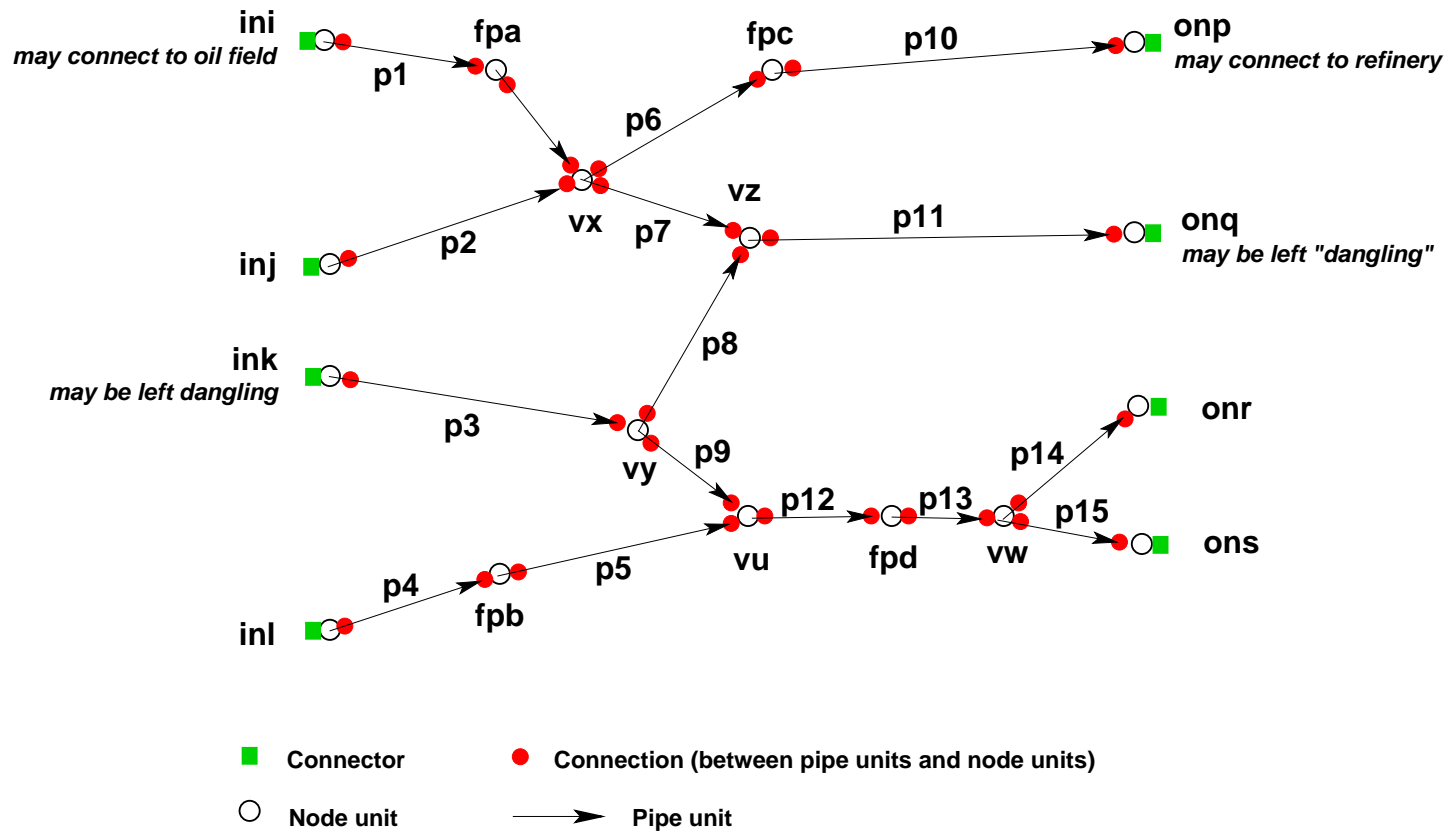


Figure 8: A Pipeline System

[3. Discussion & Interpretation, 3.2. Six Interpretations]

3.2.6. Railway Nets

Units

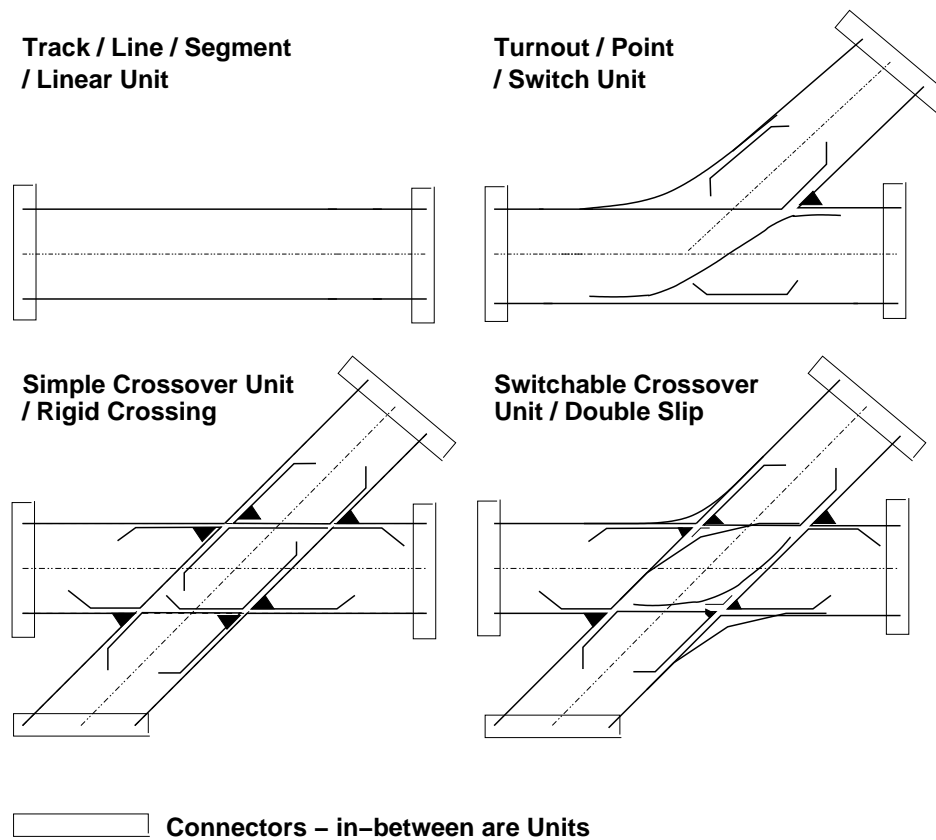


Figure 9: Four example rail units

[3. Discussion & Interpretation, 3.2. Six Interpretations, 3.2.6. Railway Nets]

An Overall Assembly

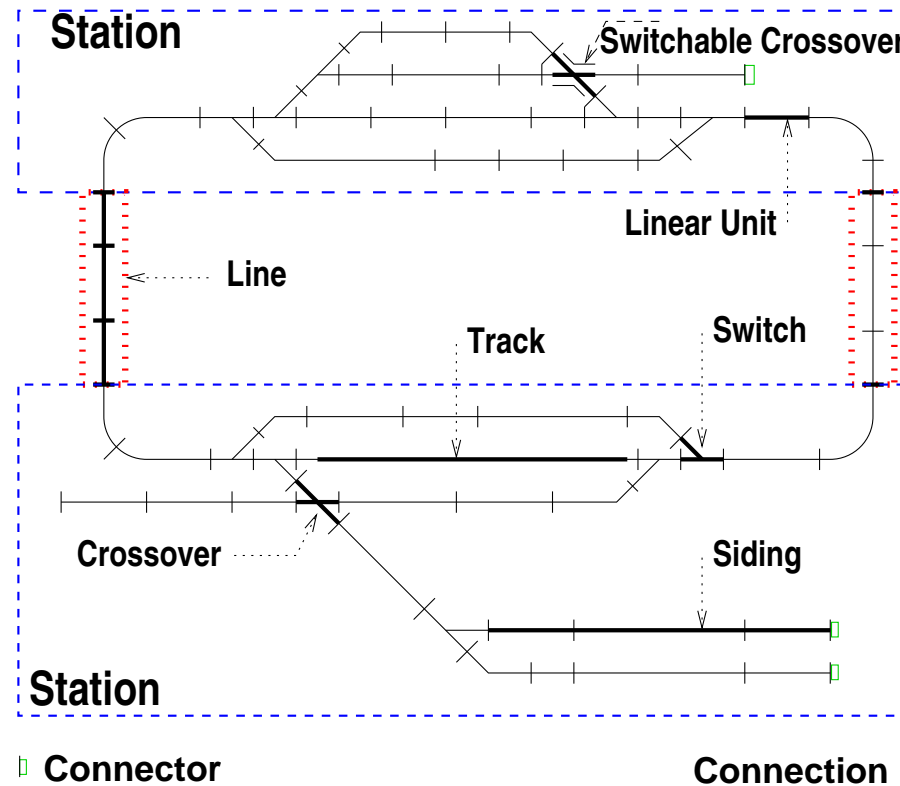


Figure 10: A “model” railway net. An Assembly of four Assemblies:
 Two stations and two lines; Lines here consist of linear rail units;
 stations of all the kinds of units shown in Fig. 9 on the preceding page.
 There were 66 connections at last count and three “dangling” connectors

[**3. Discussion & Interpretation**]**3.3. Discussion**

- It requires a somewhat more laborious effort,
 - than just “flashing” and commenting on these diagrams,
 - to show that the modelling of essential aspects of their structures
 - can indeed be done by simple instantiation
 - of the model given in the previous part of the talk.

[**3. Discussion & Interpretation**, **3.3. Discussion**]

- We can refer to a number of documents which give rather detailed domain models of
 - air traffic,
 - container line industry,
 - financial service industry,
 - health-care,
 - IT security,
 - “the market”,
 - “the” oil industry,
 - transportation nets,
 - railways, etcetera, etcetera.
- Seen in the perspective of the present paper
 - we claim that much of the modelling work done in those references
 - can now be considerably shortened and
 - trust in these models correspondingly increased.

4. Simple Entities

- The reason for our interest in ‘simple entities’
 - is that assemblies and units of systems
 - possess static and dynamic properties
 - which become contexts and states of
 - the processes into which we shall “transform” simple entities.

[**4. Simple Entities**]

4.1. Observable Phenomena

- We shall just consider ‘simple entities’.
 - By a simple entity we shall here understand
 - * a phenomenon that we can designate, viz.
 - * see, touch, hear, smell or taste, or
 - * measure by some instrument (of physics, incl. chemistry).
 - A simple entity thus has properties.
 - A simple entity is
 - * either continuous
 - * or is discrete, and then it is
 - either atomic
 - or composite.

[4. Simple Entities, 4.1. Observable Phenomena]

4.1.1. Attributes: Types and Values

- By an attribute we mean a simple property of an entity
 - a simple entity has properties p_i, p_j, \dots, p_k .
- Typically we express attributes by a pair of
 - a type designator: *the attribute is of type V* , and
 - a value: *the attribute has value v* (of type V , i.e., $v : V$).
- A simple entity may have many simple properties.
 - A continuous entity, like ‘oil’, may have the following attributes:
 - * type : *petroleum*,
 - * kind : *Brent-crude*,
 - * amount : *6 barrels*,
 - * price : *45 US \$/barrel*.

[4. Simple Entities, 4.1. Observable Phenomena, 4.1.1. Attributes: Types and Values]

— An atomic entity, like a ‘person’, may have the following attributes:

- * gender : *male*,
- * name : *Dines Bjørner*,
- * age : *71*,
- * marital status : *married*.

— A composite entity, like a railway system, may have the following attributes:

- * country : *Denmark*,
- * name : *DSB*,
- * electrified : *partly*,
- * owner : *independent public enterprise owned by Danish Ministry of Transport*.

[**4. Simple Entities**, **4.1. Observable Phenomena**]

4.1.2. Continuous Simple Entities

- A simple entity is said to be continuous
 - if it can be arbitrarily decomposed into smaller parts
 - each of which still remain simple continuous entities
 - of the same simple entity kind.
- Examples of continuous entities are:
 - oil, i.e., any fluid,
 - air, i.e., any gas,
 - time period and
 - a measure of fabric.

[**4. Simple Entities**, **4.1. Observable Phenomena**]

4.1.3. Discrete Simple Entities

- A simple entity is said to be discrete if its immediate structure is not continuous.
 - A simple discrete entity may, however, contain continuous sub-entities.
- Examples of discrete entities are:
 - persons,
 - oil pipes,
 - a railway line and
 - rail units,
 - a group of persons,
 - an oil pipeline.

[**4. Simple Entities**, **4.1. Observable Phenomena**, **4.1.3. Discrete Simple Entities**]

Atomic Simple Entities

- A simple entity is said to be atomic
 - if it cannot be meaningfully decomposed into parts
 - where these parts has a useful “value” in the context in which the simple entity is viewed and
 - while still remaining an instantiation of that entity.
- Thus a ‘physically able person’, which we consider atomic,
 - can, from the point of physical ability,
 - not be decomposed into meaningful parts: a leg, an arm, a head, etc.
- Other atomic entities could be a rail unit, an oil pipe, or a hospital bed.
- The only thing characterising an atomic entity are its attributes.

[4. Simple Entities, 4.1. Observable Phenomena, 4.1.3. Discrete Simple Entities]

Composite Simple Entities

- A simple entity, c is said to be composite
 - if it can be meaningfully decomposed
 - into sub-entities that have separate
 - meaning in the context in which c is viewed.
- Some examples of composite entities are exemplified.
 - (1) A *railway net* can be decomposed into
 - * a set of one or more *train lines* and
 - * a set of two or more *train stations*.
 - Lines and stations are themselves composite entities.

[4. Simple Entities, 1. Observable Phenomena, 3. Discrete Simple Entities, Composite Simple Entities]

- (2) An *Oil industry* whose decomposition include:
 - * one or more *oil fields*,
 - * one or more *pipeline systems*,
 - * one or more *oil refineries* and
 - * one or more *one or more oil product distribution systems*.
- Each of these sub-entities are also composite.
- Composite simple entities are thus characterisable by
 - their attributes,
 - their sub-entities, and
 - the mereology of how these sub-entities are put together.

[**4. Simple Entities**]**4.2. Mereology, Part III**

- Formula 15 on page 21 expresses that
 - whenever an individual has one proper part
 - then it has more than one.
- We mentioned there, Slide 22, that we would comment on the fact that our model appears to allow that assemblies may have just one proper part.
- We now do so.
 - We shall still allow assemblies to have just one proper part —
 - in the sense of a sub-assembly or a unit —
 - but we shall interpret the fact that an assembly always have at least one attribute.
 - Therefore we shall “generously” interpret the set of attributes of an assembly to constitute a part.

[**4. Simple Entities**, **4.2. Mereology, Part III**]

- In Sect. 5
 - we shall see how attributes of both units and assemblies of the interpreted mereology
 - contribute to the state components of the unit and assembly processes.

[**4. Simple Entities**, **4.3. Discussion**]

- In conventional modelling
 - the mereology of an infrastructure component
 - * of the kinds exemplified in Sect. 3.2
 - was modelled by modelling
 - * that infrastructure component's special mereology
 - * together, “in line”, with the modelling
 - * of unit and assembly attributes.

[**4. Simple Entities**]

4.4. Discussion

4.4.1. Modelling Simple Entities

- With the model of Sect. 2 now available
 - we do not have to model the mereological aspects,
 - but can, instead, instantiate the model of Sect. 2 appropriately.
 - We leave that to be reported upon elsewhere.
- In many conventional infrastructure component models
 - it was often difficult to separate
 - * what was mereology from
 - * what were attributes.

5. A Semantic Model of a Class of Mereologies

5.1. The Mereology Entities \equiv Processes

- The model of mereology (Slides 9–30) given earlier focused on the following simple entities
 - the assemblies,
 - the units and
 - the connectors.
- To assemblies and units we associate **CSP** processes, and
- to connectors we associate a **CSP** channels,
- one-by-one.

[**5. A Semantic Model of a Class of Mereologies**]**5.2. The ‘Calculus of Individuals’ Connections \equiv Channels**

- The connectors form the mereological attributes of the model.
- To each internal connection we associate a **CSP** channel,
 - it is “anchored” in two parts:
 - if a part is a unit then in “its corresponding” unit process, and
 - if a part is an assembly then in “its corresponding” assembly process.

[**5. A Semantic Model of a Class of Mereologies**]**5.3. Channels**

- From a system assembly we can extract all connector identifiers.
- They become indexes into an array of channels.
 - Each of the connector channel identifiers is mentioned
 - in exactly one unit or one assembly process.

[**5. A Semantic Model of a Class of Mereologies**, **5.3. Channels**]**value** $s:S$ $kis:KI\text{-set} = xtr \forall KIs(s)$ **type** $ChMap = AUI \xrightarrow{m} KI\text{-set}$ **value** $cm:ChMap = [obs_AUI(p) \mapsto obs_KIs(p) | p:P \cdot p \in xtr_Ps(s)]$ **channel** $ch[i|i:KI \cdot i \in kis] \text{ MSG}$ **5.4. Processes****5.4.1. The System Process****value** $system: S \rightarrow \text{Process}$ $system(s) \equiv assembly(s)$

[**5. A Semantic Model of a Class of Mereologies**, **5.4. Processes**]**5.4.2. The Assembly Process****value**

assembly: $a:A \rightarrow \mathbf{in, out} \{ \text{ch}[\text{cm}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_AUI}(a)) \}$ **process**

assembly(a) \equiv

$\mathcal{M}_{\mathcal{A}}(a)(\text{obs_A}\Sigma(a)) \parallel$

$\parallel \{ \text{assembly}(a') \mid a':A \cdot a' \in \text{obs_Ps}(a) \} \parallel$

$\parallel \{ \text{unit}(u) \mid u:U \cdot u \in \text{obs_Ps}(a) \}$

obs_ AΣ: $A \rightarrow A\Sigma$

$\mathcal{M}_{\mathcal{A}}$: $a:A \rightarrow A\Sigma \rightarrow \mathbf{in, out} \{ \text{ch}[\text{cm}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_AUI}(a)) \}$ **process**

$\mathcal{M}_{\mathcal{A}}(a)(a\sigma) \equiv \mathcal{M}_{\mathcal{A}}(a)(A\mathcal{F}(a)(a\sigma))$

$A\mathcal{F}$: $a:A \rightarrow A\Sigma \rightarrow \mathbf{in, out} \{ \text{ch}[\text{em}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_AUI}(a)) \} \times A\Sigma$

[5. A Semantic Model of a Class of Mereologies, 5.4. Processes]

5.4.3. Unit Processes

value

unit: $u:U \rightarrow \mathbf{in,out} \{ \text{ch}[\text{cm}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_UI}(u)) \}$ **process**

$\text{unit}(u) \equiv \mathcal{M}_{\mathcal{U}}(u)(\text{obs_U}\Sigma(u))$

$\text{obs_U}\Sigma: U \rightarrow U\Sigma$

$\mathcal{M}_{\mathcal{U}}: u:U \rightarrow U\Sigma \rightarrow \mathbf{in,out} \{ \text{ch}[\text{cm}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_UI}(u)) \}$ **process**

$\mathcal{M}_{\mathcal{U}}(u)(u\sigma) \equiv \mathcal{M}_{\mathcal{U}}(u)(U\mathcal{F}(u)(u\sigma))$

$U\mathcal{F}: U \rightarrow U\Sigma \rightarrow \mathbf{in,out} \{ \text{ch}[\text{em}(i)] \mid i:KI \cdot i \in \text{cm}(\text{obs_AUI}(u)) \}$ $U\Sigma$

[5. A Semantic Model of a Class of Mereologies]

5.5. Mereology, Part III

- A little more meaning has been added to the notions of parts and connections.
- The **within** and **adjacent to** relations between parts (assemblies and units) reflect a phenomenological world of geometry, and
- the **connected** relation between parts (assemblies and units)
 - reflect both physical and conceptual world understandings:
 - * physical world in that, for example, radio waves cross geometric “boundaries”, and
 - * conceptual world in that ontological classifications typically reflect lattice orderings where *overlaps* likewise cross geometric “boundaries”.

[**5. A Semantic Model of a Class of Mereologies**]

5.6. Discussion

- That completes our ‘contribution’:
 - A mereology of systems has been given
 - a syntactic explanation, Sect. 2,
 - a semantic explanation, Sect. 5 and
 - their relationship to classical mereologies.

6. Conclusion

6.1. Summary

- We have proposed a simple model which we claim captures a large variety of structures of societal infrastructure components. The model focused on **parts** and **connections** between parts.
- We have, rather briefly, held that model up against a variety of diagrammatic renditions of specific societal infrastructure components and claimed that the model is relevant for their formalisation.
- We have finally shown how one can relate simple entities to **CSP processes** and connectors to **CSP channels**.

[**6. Conclusion**]**6.2. What Have We Achieved ?**

- There is, as we indicated a bewildering variety of from societal infrastructure component to “gadget” structures – and these structures must be modelled.
- We claim that the mereology model provides a common denominator for all of these: that the model is generic and can be simply instantiated for each of the shown, and, we again claim, many other domain examples.
- We claim that the model can serve as a basis for investigating the axiom systems proposed for mereology (Casati&Varzi 1999) and a calculus of individuals (Bowman&Clarke 1981).
- We thus claim to have a simple model for the kind of mereologies presented in the literature.

[**6. Conclusion**]

6.3. Open Points

- We have yet to carefully demonstrate two classes of things:
 - (i) to properly refine our mereology model into models for the sub-entity structures of specific societal infrastructure components etc.; and
 - (ii) to identify the exact relations between our model of mereology and the axiom systems presented in the literature.

7. Acknowledgements

- I thank University of Saarland for hosting me during some of the time when I wrote this paper.
- And I thank Prof. Wolfgang Reisig and his colleagues for allowing me to present this work-in-progress.