Mereologies in Computing Science

Uppsala: Thursday, 11 November 2010

Dines Bjørner

1. Abstract

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).
- Mereology is the study, knowledge and practice of part-hood relations:
  - of the relations of part to whole and
  - the relations of part to part within a whole.
- By parts we shall here understand simple entities — of the kind illustrated in this talk.

- Manifest simple entities of domains
  - are either continuous (fluid, gaseous)
  - or discrete (solid, fixed), and if the latter, then
    * either atomic
    * or composite.
  - It is how the sub-entities of a composite entity
    * are "put together"
    * that "makes up" a mereology of that composite entity

    — at least such as we shall study the mereology concept.
- In this talk we shall study some ways of modelling the mereology of composite entities.

- One way of modelling mereologies is using
  - **sorts,**
  - **observer functions** and
  - **axioms** (McCarthy style),
- another is using **CSP**.

2. Introduction

2.1. Physics and Societal Infrastructures

2.1.1. Physics

- **Physicists** 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.

- **Domain engineers** 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.

2.1.2. In Nature

- **Physicists** 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.

- **Domain engineers** 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" !

- 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.

2.2. 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.
- We informally show that the assembly/unit structures indeed model structures of a variety of infrastructure components.
- 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 and now
    * of semantic aspects of a class of mereologies.

3. A Syntactic Model of a Class of Mereologies

3.1. 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 | U
value
  obs_Ps: (S|A) → P-**set**

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

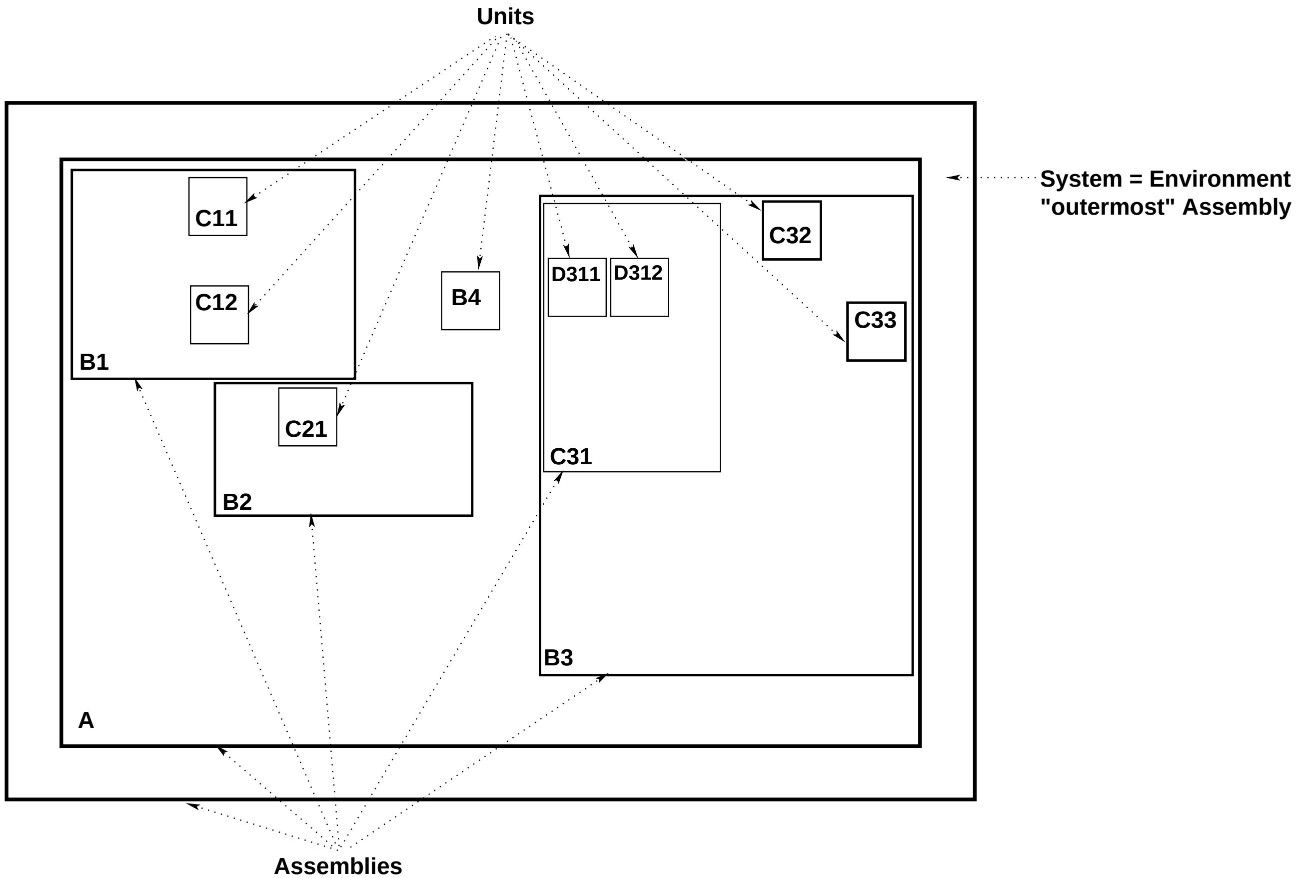

Figure 1: Assemblies and Units “embedded” in an Environment

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

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

value

$\text{xtr\_Ps}: (\text{S}|\text{A}) \rightarrow \text{P-}\mathbf{set}$

$\text{xtr\_Ps}(\text{a}) \equiv$

$\quad \mathbf{let}\ \text{ps} = \{\text{a}\} \cup \text{obs\_Ps}(\text{a})\ \mathbf{in}\ \text{ps} \cup \mathbf{union}\{\text{xtr\_Ps}(\text{a}')|\text{a}':\text{A}\cdot\text{a}' \in \text{ps}\}\ \mathbf{end}$

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

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

type
  AUI
value
  obs_AUI: P $\rightarrow$ AUI
axiom
  $\forall$ a:A $\cdot$
    let ps = obs_Ps(a) **in**
    $\forall$ p′,p″:P $\cdot$ {p′,p″}$\subseteq$ps $\wedge$ p′$\neq$p″ $\Rightarrow$ obs_AUI(p′)$\neq$obs_AUI(p″) $\wedge$
    $\forall$ a′,a″:A $\cdot$ {a′,a″}$\subseteq$ps $\wedge$ a′$\neq$a″ $\Rightarrow$ xtr_Ps(a′)$\cap$ xtr_Ps(a″)={} **end**

3.2. ‘Adjacency’ and ‘Within’ Relations

- 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

$$\text{i\_next\_to: P} \times \text{P} \rightarrow \text{A} \xrightarrow{\sim} \textbf{Bool}, \textbf{ pre } \text{i\_next\_to(p,p}')\text{(a): p} \neq \text{p}'$$

$$\text{i\_next\_to(p,p}')\text{(a)} \equiv \exists\ \text{a}'\text{:A} \cdot \text{a}'\text{=a} \vee \text{a}' \in \text{xtr\_Ps(a)} \cdot \{\text{p,p}'\} \subseteq \text{obs\_Ps(a}')$$

- One part, p, is said to be *immediately within* another part, p′in an assembly a
  - if there exists an assembly, a′ equal to or embedded in a
  - such that p is observable in a′.

value

$$\text{i\_within: } P \times P \to A \xrightarrow{\sim} \textbf{Bool}$$

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

$$\exists\, a'{:}A \cdot\ (a{=}a' \vee a' \in \text{xtr\_Ps}(a)) \cdot p'{=}a' \wedge p \in \text{obs\_Ps}(a')$$

- We can generalise the immediate 'within' property.
- A part, p, is (transitively) within a part p′, 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 within(p′′,p)(a).

value

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

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

$$\quad \text{i\_within}(p,p')(a) \vee \exists\, p'':P \cdot p'' \in \text{obs\_Ps}(p) \wedge \text{within}(p'',p')(a)$$

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

value

$\text{within}'\colon \text{P} \times \text{P} \to \text{A} \xrightarrow{\sim} \textbf{Bool}$

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

$\quad \text{i\_within}(\text{p},\text{p}')(\text{a}) \vee \exists\, \text{p}''\text{:P} \cdot \text{p}'' \in \text{xtr\_Ps}(\text{p}) \wedge \text{i\_within}(\text{p}'',\text{p}')(\text{a})$

lemma: $\text{within} \equiv \text{within}'$

3.2.1. 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

$$
\begin{array}{l}
\text{adjacent: } P \times P \to A \xrightarrow{\sim} \textbf{Bool} \\
\text{adjacent}(p,p')(a) \equiv \\
\quad \text{i\_next\_to}(p,p')(a) \vee \\
\quad \exists\, p'',p''':P \cdot \{p'',p'''\} \subseteq \text{xtr\_Ps}(a) \wedge \text{i\_next\_to}(p'',p''')(a) \wedge \\
\qquad ((p=p'') \vee \text{within}(p,p'')(a)) \wedge ((p'=p''') \vee \text{within}(p',p''')(a))
\end{array}
$$

3.3. Mereology, Part I

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

$$\forall x(x \sqsubseteq x)^1 \tag{1}$$
$$\forall x, y(x \sqsubseteq y) \wedge (y \sqsubseteq x) \Rightarrow (x = y) \tag{2}$$
$$\forall x, y, z(x \sqsubseteq y) \wedge (y \sqsubseteq z) \Rightarrow (x \sqsubseteq z) \tag{3}$$

- 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.

$$\forall x \sqsubset y =_{\text{def}} x(x \sqsubseteq y) \wedge \neg(x = y) \tag{4}$$
$$\forall\forall x, y(x \sqsubseteq y) \quad \Leftrightarrow \quad (x \sqsubset y) \vee (x = y) \tag{5}$$

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

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

$$\forall x \neg (x \sqsubset x) \tag{6}$$
$$\forall x, y (x \sqsubset y) \Rightarrow \neg (y \sqsubset x) \tag{7}$$
$$\forall x, y, z (x \sqsubset y) \wedge (y \sqsubset z) \Rightarrow (x \sqsubset z) \tag{8}$$

- *Overlap*, $\bullet$, is also a relation of parts:
  – Two individuals overlap if they have parts in common:

$$x \bullet y =_{\text{def}} \exists z (z \sqsubset x) \wedge (z \sqsubset y) \tag{9}$$
$$\forall x (x \bullet x) \tag{10}$$
$$\forall x, y (x \bullet y) \Rightarrow (y \bullet x) \tag{11}$$

- Proper overlap, $\circ$, can be defined:

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

- 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.

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

- Disjointness is symmetric:

$$\forall x, y (x \oint y) \Rightarrow (y \oint x) \tag{14}$$

- 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.

$$\forall x, y (y \sqsubset x) \;\; \Rightarrow \;\; \exists z (z \sqsubset x) \land (z \oint y) \tag{15}$$

- Formulas 1–3 and 15 together determine the *minimal mereology*, $\mathcal{M}_{\mathcal{M}\text{inimal}}$.
- 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 of parts having attributes.

3.4. A Syntactic Model of a Class of Mereologies

3.4.1. 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.

- Connectors provide for connections between parts.
- A connector is an ability be 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.*

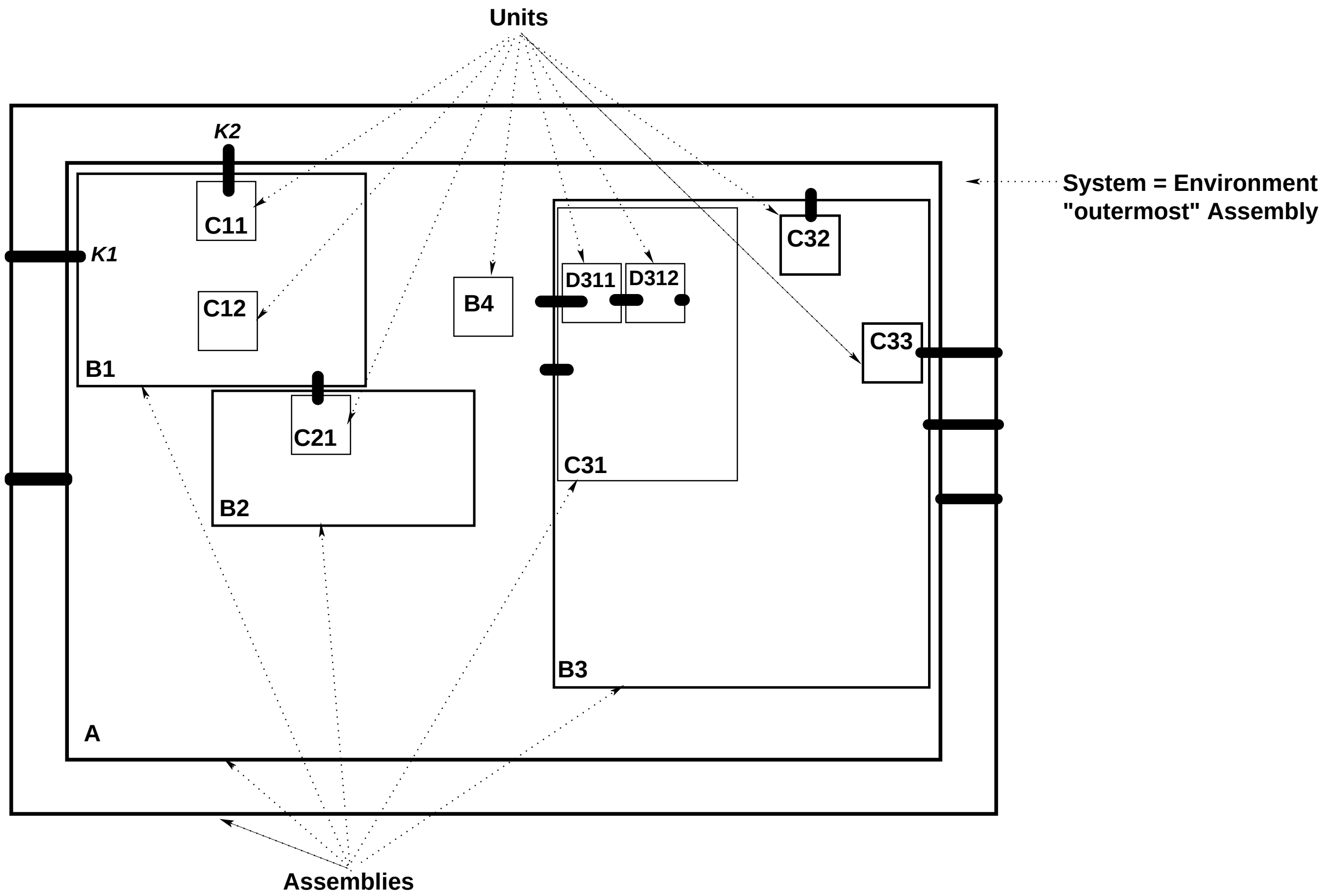

Figure 2: Assembly and Unit Connectors: Internal and External

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

- Figure 2 on the facing page "adds" connectors to Fig. 1 on page 14.
- The idea is that connectors
  - allow an assembly to be connected to any embedded part, and
  - allow two adjacent parts to be connected.
- In Fig. 2 on the facing page
  - the environment is connected, by *K2*, to part C11;
  - the "external world" is connected, by K1, to B1;
  - etcetera.

- 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.

type

K

value

obs_Ks: S → K-**set**

obs_KI: K → KI

obs_Is: K → AUI-**set**

obs_KIs: P → KI-**set**

axiom

∀ k:K • **card** obs_Is(k)=2,

∀ s:S,k:K • k ∈ obs_Ks(s) ⇒

  ∃ p:P • p ∈ xtr_Ps(s) ⇒ obs_AUI(p) ∈ obs_Is(k),

∀ s:S,p:P • ∀ ki:KI • ki ∈ obs_KIs(p) ⇒

  ∃! k:K • k ∈ obs_Ks(s) ∧ ki=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.

- We need define an auxiliary function.
  - xtr∀KIs(p) applies to a system
  - and yields all its connector identifiers.

value

xtr∀KIs: S → KI-**set**

xtr∀Ks(s) ≡ {obs_KI(k)|k:K•k ∈ obs_Ks(s)}

3.5. 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.

- 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.

- We earlier introduced the following operators:
  - $\sqsubseteq$, $\sqsubset$, $\bullet$, $\circ$, and $\oint$
- In some of the mereology literature [BowmanLClarke81, BowmanLClarke85, CasatiVarzi1999] these operators are symbolised with caligraphic letters:
  - $\sqsubseteq$: $\mathcal{P}$: part,
  - $\sqsubset$: $\mathcal{PP}$: proper part,
  - $\bullet$: $\mathcal{O}$: overlap and
  - $\oint$: $\mathcal{U}$: underlap.

4. 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).

4.1. 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.

4.2. 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.

4.2.1. Air Traffic

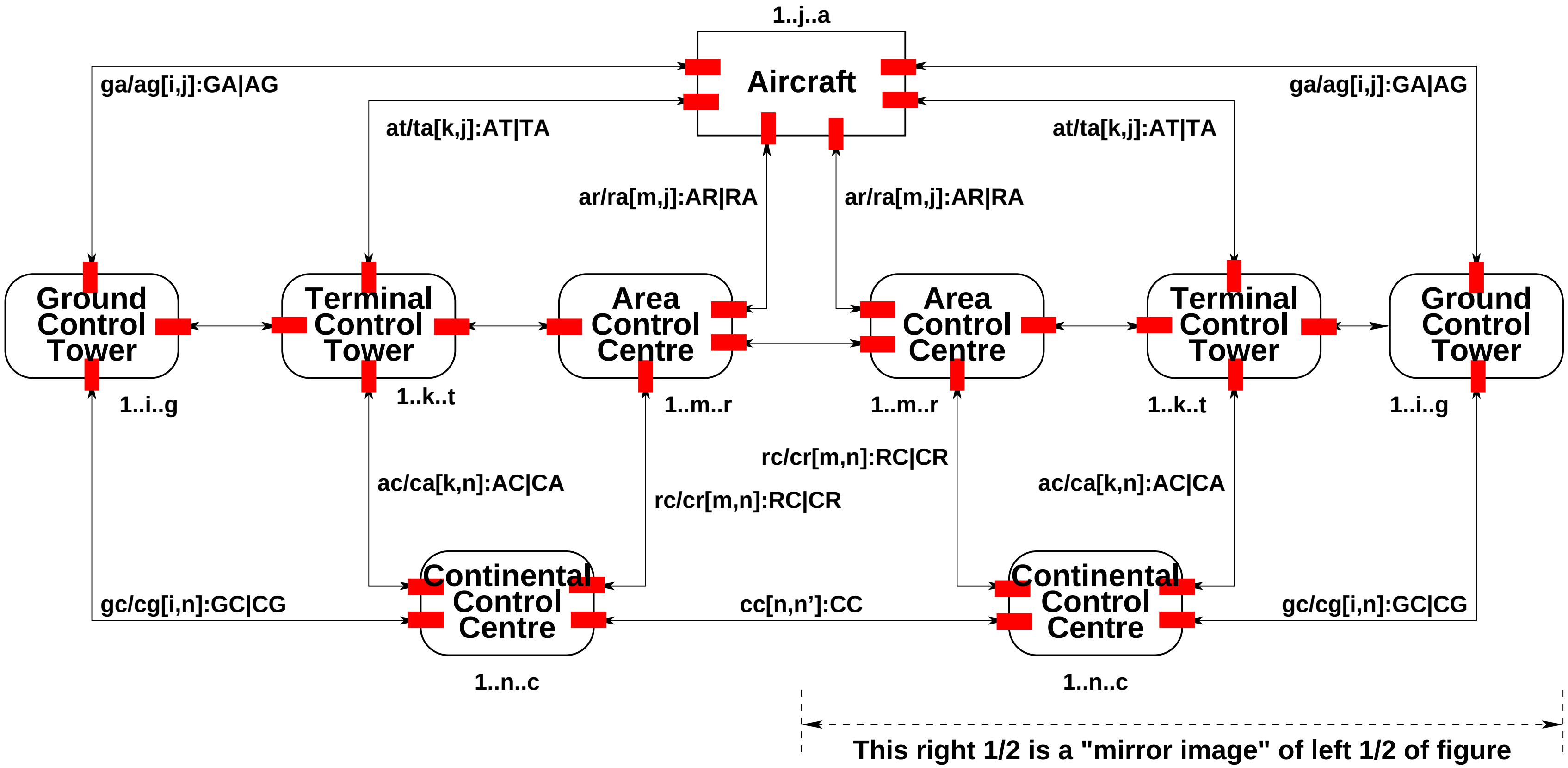

Figure 3: An air traffic system. Black boxes and lines are units; **red** **boxes** are connections

4.2.2. Buildings

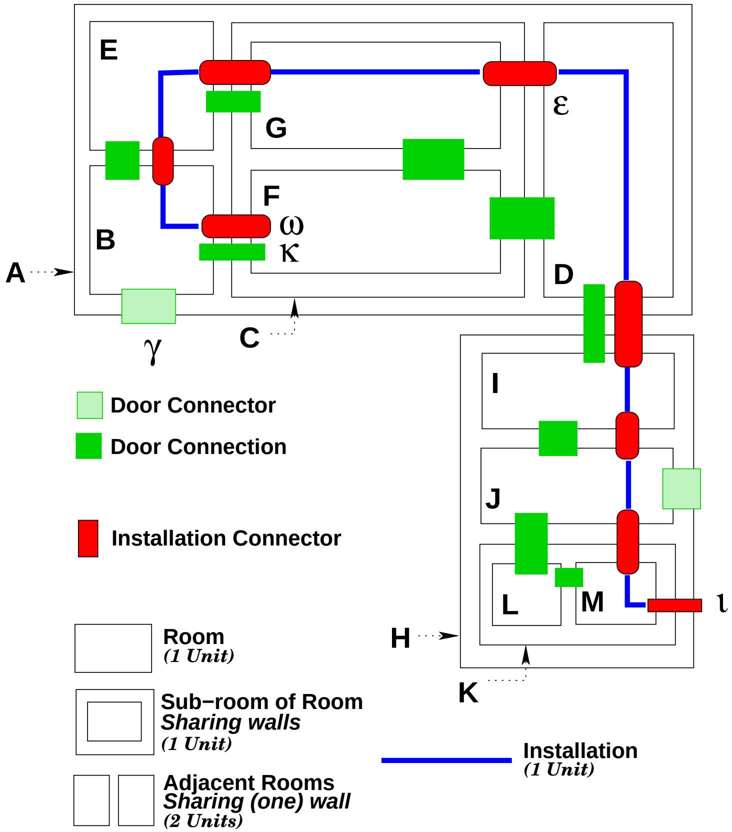

Figure 4: A building plan with installation

4.2.3. Financial Service Industry

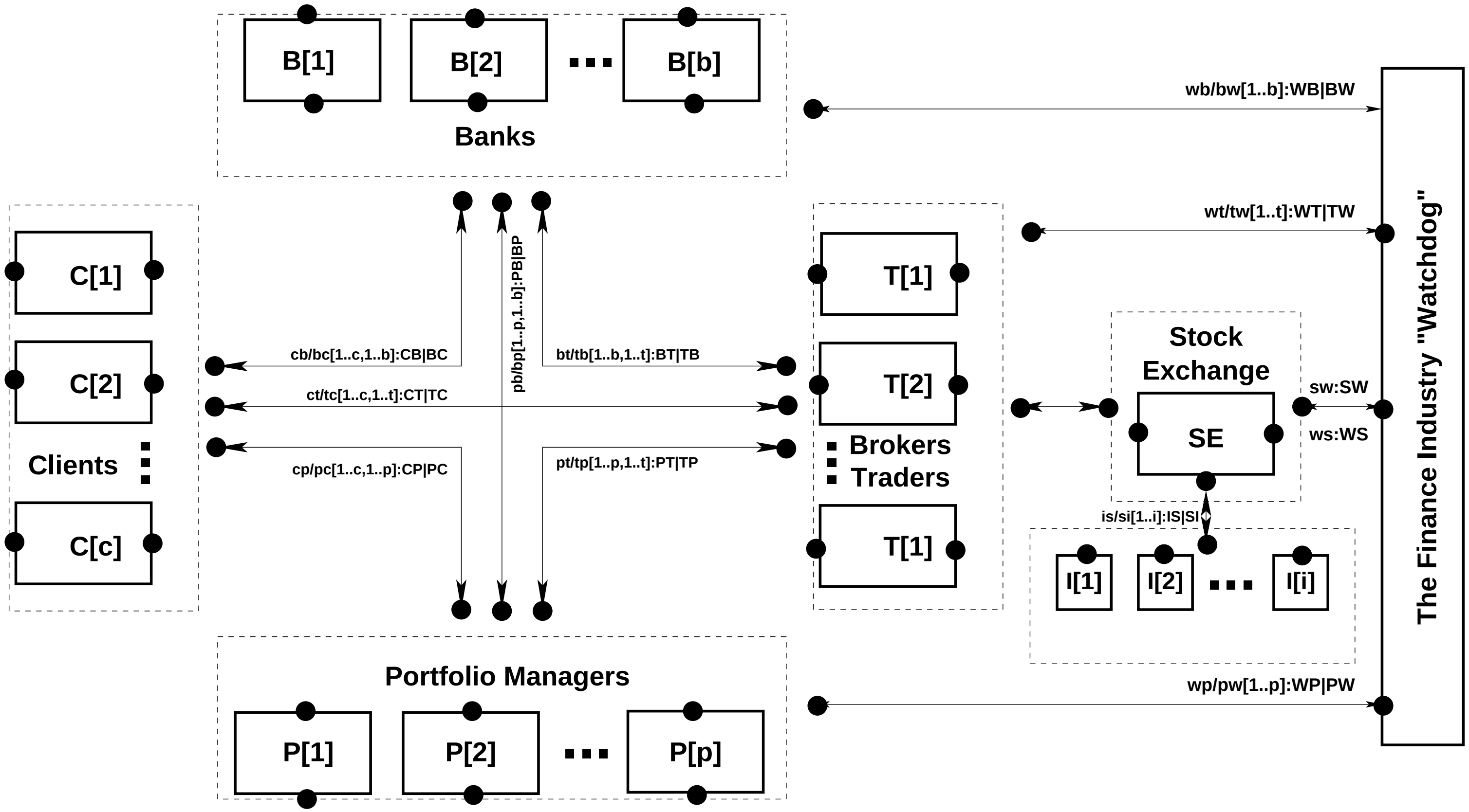

Figure 5: A financial service industry

4.2.4. Machine Assemblies

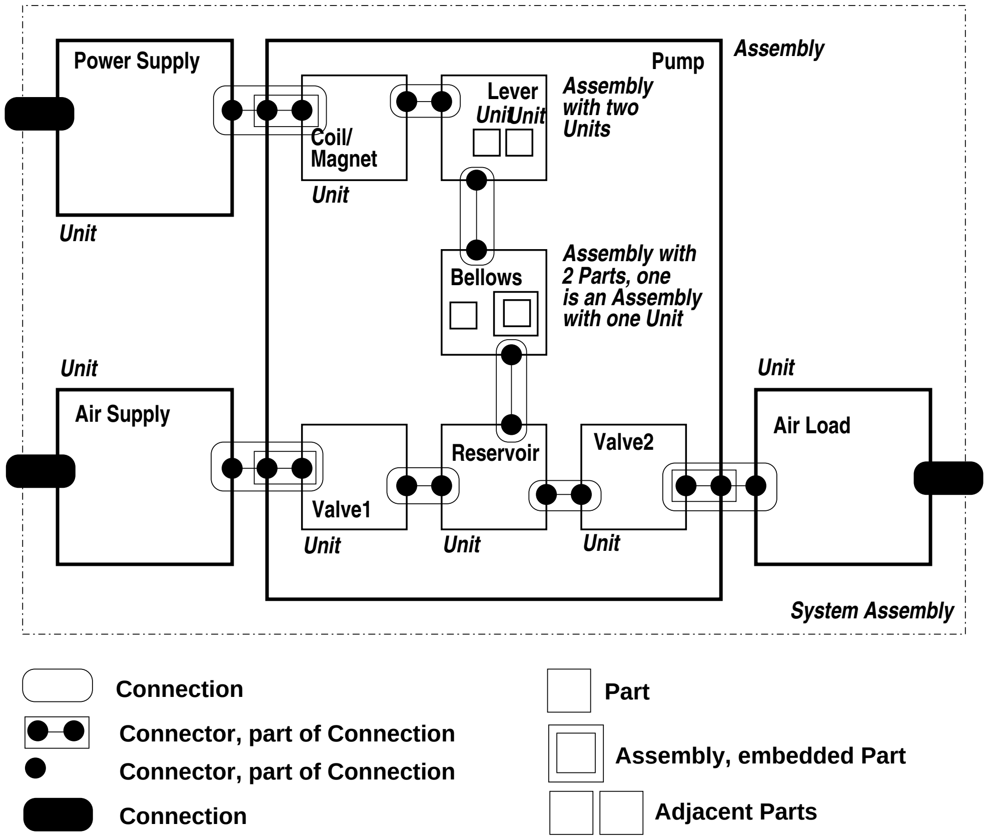

Figure 6: An air pump, i.e., a physical mechanical system

4.2.5. Oil Industry

4.2.5.1. “The” Overall Assembly

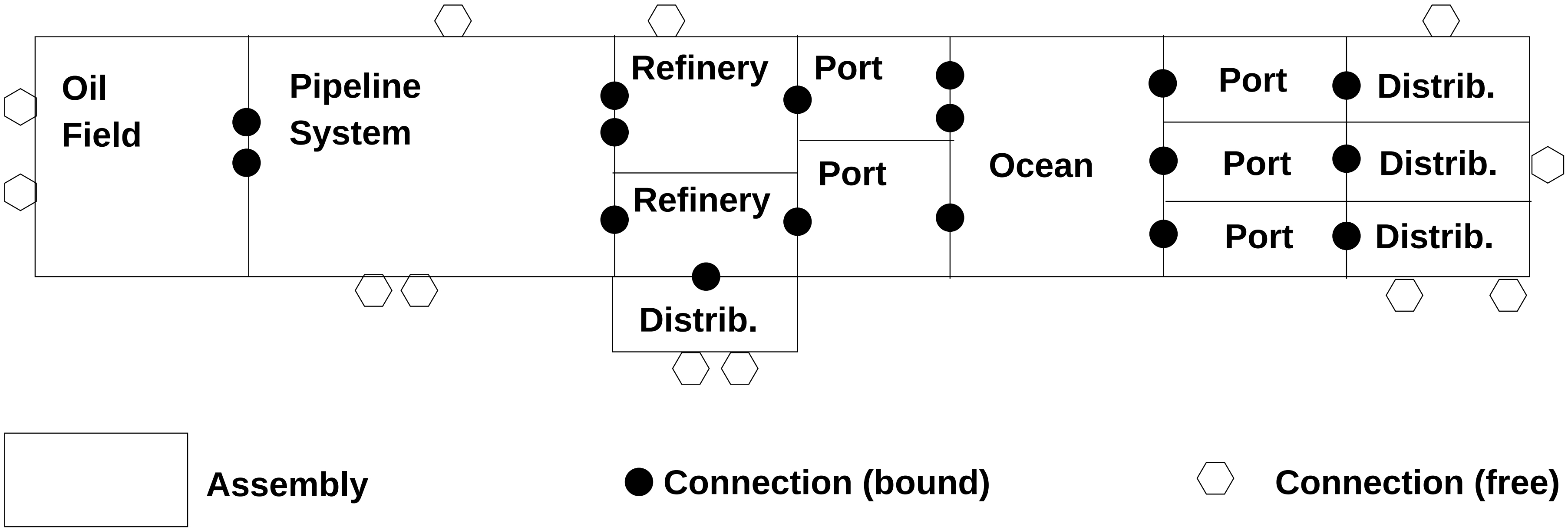

Figure 7: A Schematic of an Oil Industry

4.2.5.2. A Concretised Assembly Unit

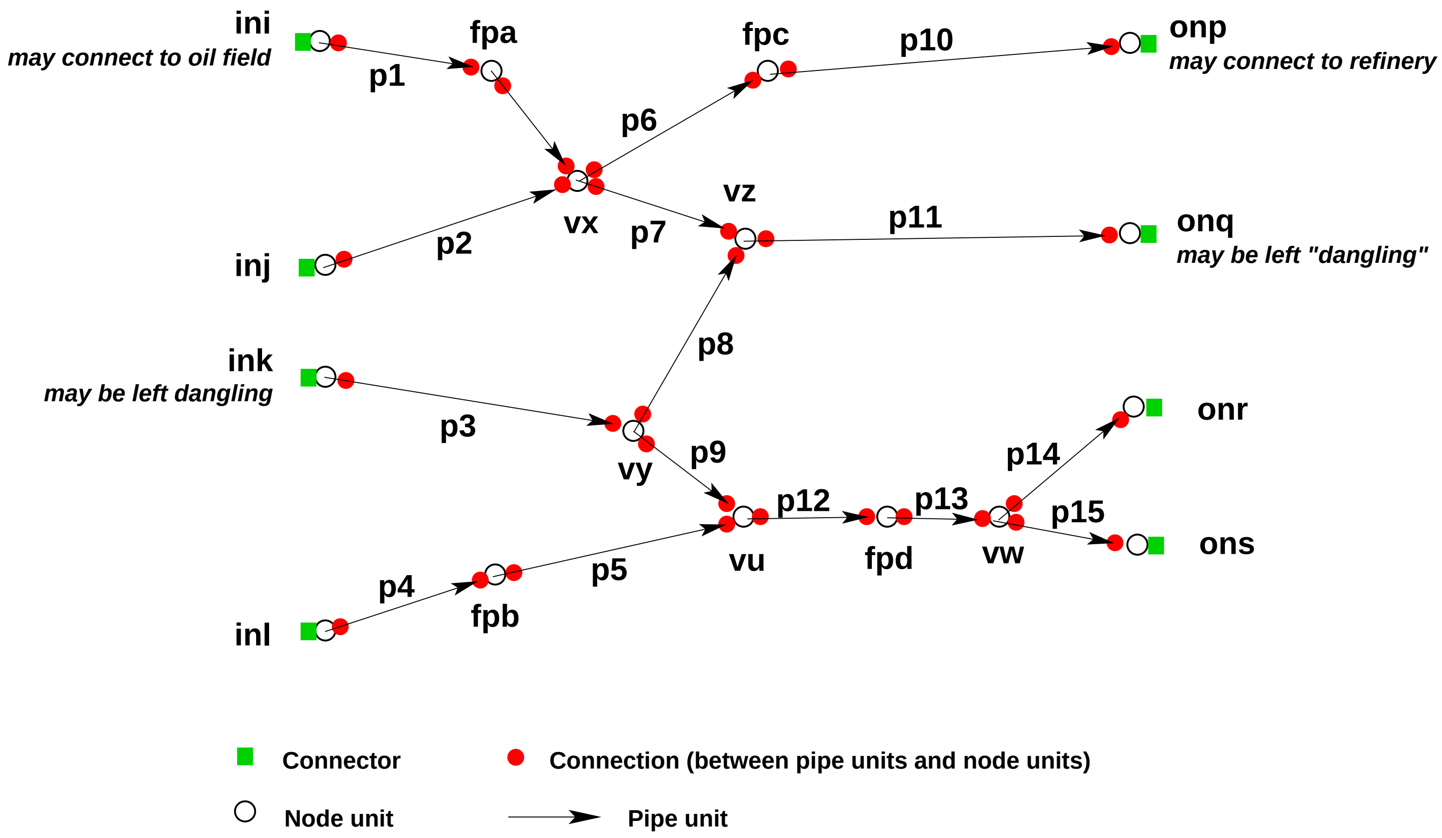

Figure 8: A Pipeline System

4.2.6. Railway Nets

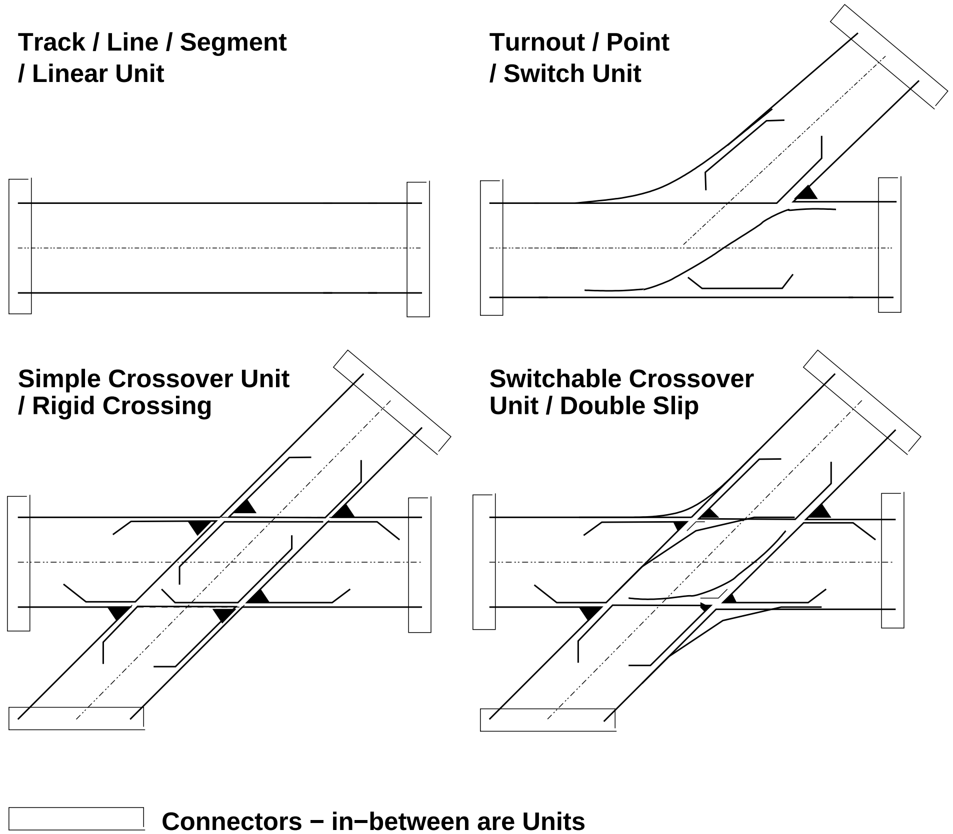

Figure 9: Four example rail units

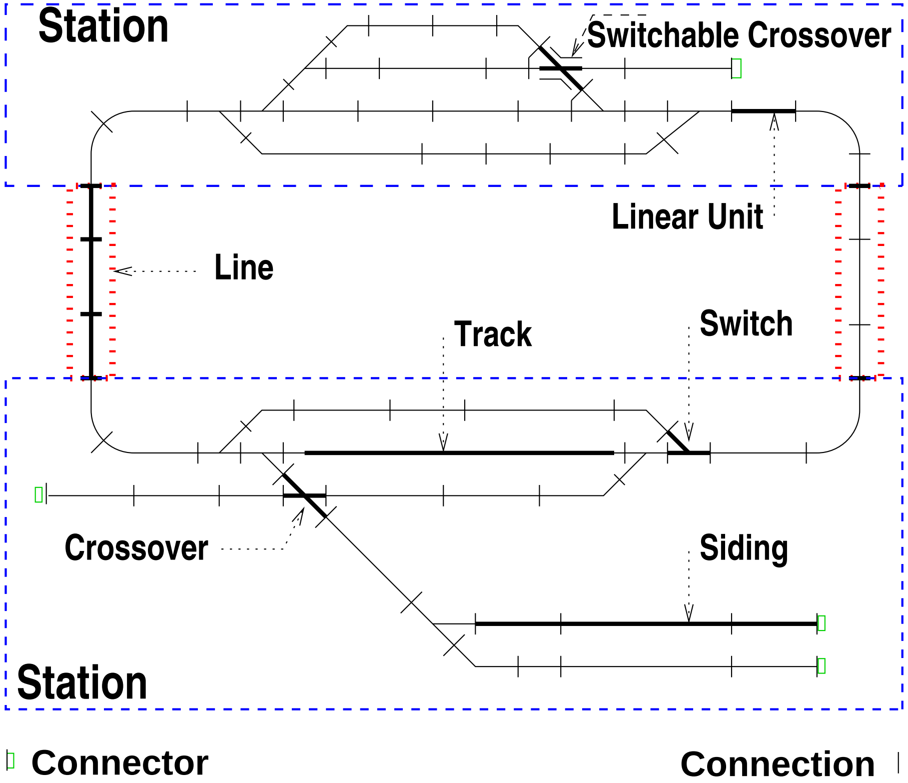

Figure 10: A "model" railway net. An Assembly of four Assemblies:
Two stations and two lines; Lines here consist of linear rail units;
stations of all the kinds of units shown in Fig. 9 on the facing page.
There are 66 connections and four "dangling" connectors

4.3. 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.

- 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[2],
  - transportation nets[3],
  - 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.

[2] http://www2.imm.dtu.dk/~db/pipeline.pdf
[3] http://www2.imm.dtu.dk/~db/transport.pdf

4.4. Mereology, Part III

- Formula 15 on page 26 expresses that
  - whenever an individual has one proper part
  - then it has more than one.
- We mentioned there, Slide 27, 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.

- 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.

4.5. Discussion

- In Sect. 3.2 we interpreted the model of mereology in six examples.
- The units of Sect. 2
  - which in that section were left uninterpreted
  - now got individuality —
    * in the form of
      · aircraft,
      · building rooms,
      · rail units and
      · oil pipes.
  - Similarly for the assemblies of Sect. 2. They became
    * pipeline systems,
    * oil refineries,
    * train stations,
    * banks, etc.

- 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.
- 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.

5. A Semantic Model of a Class of Mereologies

5.1. The Mereology Entities ≡ Processes

- The model of mereology (Slides 13–38) given earlier focused on the following simple entities (i) the assemblies, (ii) the units and (iii) the connectors.
- To assemblies and units we associate `CSP` processes, and
- to connectors we associate a `CSP` channels,
- one-by-one.
- The connectors form the mereological attributes of the model.

5.1.1. Channels

- The CSP channels,
  - are each "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.
- 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 two unit or assembly processes.

value

s:S

kis:KI-**set** = xtr∀KIs(s)

type

ChMap = AUI $\overrightarrow{m}$ KI-**set**

value

cm:ChMap = [obs_AUI(p)↦obs_KIs(p)|p:P•p ∈ xtr_Ps(s)]

channel

ch[i|i:KI•i ∈ kis] MSG

5.2. Process Definitions

value

system: S → **Process**
system(s) ≡ assembly(s)

assembly: a:A→**in**,**out** {ch[cm(i)]|i:KI•i ∈ cm(obs_AUI(a))} **process**
assembly(a) ≡
  $\mathcal{M}_{\mathcal{A}}$(a)(obs_AΣ(a)) ∥
  ∥ {assembly(a′)|a′:A•a′ ∈ obs_Ps(a)} ∥
  ∥ {unit(u)|u:U•u ∈ obs_Ps(a)}
obs_AΣ: A → AΣ

$\mathcal{M}_{\mathcal{A}}$: a:A→AΣ→**in**,**out** {ch[cm(i)]|i:KI•i ∈ cm(obs_AUI(a))} **process**
$\mathcal{M}_{\mathcal{A}}$(a)(a$\sigma$) ≡ $\mathcal{M}_{\mathcal{A}}$(a)($A\mathcal{F}$(a)(a$\sigma$))

A$\mathcal{F}$: a:A → AΣ → **in**,**out** {ch[em(i)]|i:KI•i ∈
cm(obs_AUI(a))}×AΣ

unit: u:U $\rightarrow$ **in**,**out** $\{$ch[cm(i)]$|$i:KI•i $\in$ cm(obs_UI(u))$\}$ **process**
unit(u) $\equiv$ $\mathcal{M}_{\mathcal{U}}$(u)(obs_U$\Sigma$(u))
obs_UΣ: U $\rightarrow$ UΣ

$\mathcal{M}_{\mathcal{U}}$: u:U $\rightarrow$ UΣ $\rightarrow$ **in**,**out** $\{$ch[cm(i)]$|$i:KI•i $\in$ cm(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 $\rightarrow$ UΣ $\rightarrow$ **in**,**out** $\{$ch[em(i)]$|$i:KI $\cdot$ i $\in$ cm(obs_AUI(u))$\}$ UΣ

5.3. 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”.

5.4. 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.

5.5. Acknowledgements

- I thank Lars-Henrik Eriksson and his colleagues for inviting me to give a two week "PhD School" series of lectures at Uppsala.
- And I also thank for the opportunity to give this seminar.