Many-sorted logic

Jouko Väänänen$^{1,2}$

$^1$Department of Mathematics and Statistics, University of Helsinki

$^2$Institute for Logic, Language and Computation, University of Amsterdam

January 2014
Introduction
Several “sorts" of variables
- elements and sequences of elements
- vector spaces: scalars and vectors
- geometry: points and lines
- second order logic: individuals and subsets.
Many-sorted logic

- Can be translated into one-sorted logic
- Many-sorted version of a theorem is sometimes more powerful than the single sorted version.
- Craig Interpolation Theorem.
- Applications of the many-sorted interpolation theorem.
- Applications to "symbiosis" between set theory and model theory.
Single sorted logic
An ordinary single sorted first order structure (model) \( \mathcal{M} = (M, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l) \) consists of

1. Universe \( M \neq \emptyset \)
2. Relations \( R_1, \ldots, R_m \) between elements of the universe
3. Functions \( f_1, \ldots, f_k \) between elements of the universe
4. Distinguished constants \( c_1, \ldots, c_l \) in the universe.
where ...

\[ R_i \subseteq M \times \ldots \times M \]

\[ f_i : M \times \ldots \times M \to M \]

\[ c_i \in M \]
Syntax in single sorted logic

\[ t = t' \]
\[ Rt_1 \ldots t_n \]
\[ \neg \phi \]
\[ \land_n \phi_n \]
\[ \lor_n \phi_n \]
\[ \exists x_i \phi \]
\[ \forall x_i \phi \]

In **second order logic** add

\[ \exists X_i^m \phi \text{ where } X_i^m \text{ is } m\text{-ary} \]
\[ \forall X_i^m \phi \text{ where } X_i^m \text{ is } m\text{-ary} \]

In **monadic** second order logic \( m = 1 \).
Assignments $\nu$ into $\mathcal{M}$ are functions on $\text{voc}(L)$-variables such that $\nu(x_i) \in M$ for each $x_i$. Modified assignment $\nu(a/x_i)$ as usual.

**Definition**

**First order logic:**

$$\mathcal{M} \models _{\nu} \exists x_i \phi \iff \mathcal{M} \models _{\nu(a/x_i)} \phi \text{ for some } a \in M.$$  
$$\mathcal{M} \models _{\nu} \forall x_i \phi \iff \mathcal{M} \models _{\nu(a/x_i)} \phi \text{ for all } a \in M.$$  

**Second order logic:**

$$\mathcal{M} \models _{\nu} \exists X_i^m \phi \iff \mathcal{M} \models _{\nu(A/X_i^m)} \phi \text{ for some } A \subseteq M^m.$$  
$$\mathcal{M} \models _{\nu} \forall X_i^m \phi \iff \mathcal{M} \models _{\nu(a/X_i^m)} \phi \text{ for all } A \subseteq M^m.$$
Assignments $\nu$ into $\mathcal{M}$ are functions on $\mathrm{voc}(L)$-variables such that $\nu(x_i) \in M$ for each $x_i$. Modified assignment $\nu(a/x_i)$ as usual.

**Definition**

First order logic:

\[
\mathcal{M} \models \nu \exists x_i \phi \iff \mathcal{M} \models \nu(a/x_i) \phi \quad \text{for some } a \in M.
\]

\[
\mathcal{M} \models \nu \forall x_i \phi \iff \mathcal{M} \models \nu(a/x_i) \phi \quad \text{for all } a \in M.
\]

Second order logic:

\[
\mathcal{M} \models \nu \exists X^m_i \phi \iff \mathcal{M} \models \nu(A/x^m_i) \phi \quad \text{for some } A \subseteq M^m.
\]

\[
\mathcal{M} \models \nu \forall X^m_i \phi \iff \mathcal{M} \models \nu(a/x^m_i) \phi \quad \text{for all } A \subseteq M^m.
\]
Many sorted logic: its structures
A many-sorted structure (model) $\mathcal{M} = (\{M_1, \ldots, M_n\}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l)$ consists of:

1. Universes ("sorts") $M_1, \ldots, M_n$, each $\neq \emptyset$
2. Relations $R_1, \ldots, R_m$ between elements of the universes
3. Functions $f_1, \ldots, f_k$ between elements of the universes
4. Distinguished constants $c_1, \ldots, c_l$ in the universes.

We call $\mathcal{M}$ $n$-sorted. $\mathcal{M}$ is "strict" if the $M_i$ are disjoint, otherwise "lax". We allow $n = 0$, i.e. "no sorts" as a special case.
where ...

\[ R_i \subseteq M_{i_1} \times \ldots \times M_{i_s} \]
\[ s(R_i) = \langle i_1, \ldots, i_s \rangle \subseteq \{1, \ldots, n\} \]

\[ f_i : M_{i_1} \times \ldots \times M_{i_s} \rightarrow M_r \]
\[ s(f_i) = \langle i_1, \ldots, i_s, r \rangle \subseteq \{1, \ldots, n\} \]

\[ c_i \in M_j \]
\[ s(c_i) = j \in \{1, \ldots, n\} \]

No relation between elements of sort \( s \) or sort \( s' \) and elements of sort \( s'' \).
Example

An ordinary single-sorted first order structure

$$(M, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l)$$

resurrects in this notation as a 1-sorted structure

$$(\{M\}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l)$$
A Chu space is a strict 2-sorted structure \((\{A, X\}, R)\), (denoted \((A, X, R)\)), where \(R \subseteq A \times X\). For example:

<table>
<thead>
<tr>
<th>(A)</th>
<th>(X)</th>
<th>(Rax)</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>sets</td>
<td>(a \in x)</td>
</tr>
<tr>
<td>models</td>
<td>sentences</td>
<td>(a \models x)</td>
</tr>
<tr>
<td>sentences</td>
<td>models</td>
<td>(x \models a).</td>
</tr>
</tbody>
</table>
A vector space is a strict 2-sorted structure

\[ V = (\{F, V\}, +_F, \cdot_F, 0_F, 1_F, +_V, \cdot_V, 0_V) , \]

where

1. \((F, +_F, \cdot_F, 0_F, 1_F)\) is a field
2. \((V, +_V, 0_V)\) is an Abelian group
3. \(\cdot_V : F \times V \to V\) satisfies
   - \(a \cdot_V (u +_V w) = a \cdot_V u +_V a \cdot_V w\)
   - \((a \cdot_F b) \cdot_V u = a \cdot_V (b \cdot_V u)\)
   - \((a +_F b) \cdot_V u = a \cdot_V u +_V b \cdot_V u\)
   - \(1_F \cdot_V u = u\)
A vector space is a strict 2-sorted structure

\[ \mathcal{V} = (F, V, +, \cdot, 0), \]

where

1. \( F \) is a field
2. \((V, +, 0)\) is an Abelian group
3. The function \((a, v) \mapsto av(= a \cdot v) : F \times V \to V\) satisfies
   - \( a(u + w) = au + aw \)
   - \((ab) \cdot u = a(bu)\)
   - \((a + b)u = au + bu\)
   - \(1u = u\)
Model with a pairing function

Example

A first order structure with a pairing function is a strict 2-sorted structure

\[ \mathcal{M} = (\{N, M\}, \pi, \ldots), \]

where \( \pi \) is a pairing function, i.e. a bijection \( M \times M \to N \). We can also let \( N = M^2 \) and \( \pi(x, y) = (x, y) \).

Note: If \( M \) is finite, \( \pi \) cannot be a function into \( M \), so a new sort is needed.
A first order structure with **finite sequences** is a strict 3-sorted structure

$$\mathcal{M} = (\bigcup_{n} M^n, \mathbb{N}, M, \text{len}, 0, s, \sigma...),$$

where $\text{len}(a) = n$ if $a \in M^n$, $s$ is the successor function on $\mathbb{N}$, and

$$\sigma(\langle a_1, \ldots, a_n \rangle, i) = a_i.$$
A first order structure with \textit{second order domain} is a strict 2-sorted structure

\[
M = (\{\mathcal{P}(M), M\}, \in, \ldots),
\]

where

1. $\in$ is the membership relation $a \in B$ between $a \in M$ and $B \subseteq M$.
2. $(M, \ldots)$ is a first order structure.
Higher order domains

Example

A first order structure with second and third order domains is a strict 3-sorted structure

\[ \mathcal{M} = (\{ \mathcal{P}(\mathcal{P}(\mathcal{M})), \mathcal{P}(\mathcal{M}), \mathcal{M} \}, \in_1, \in_2, \ldots ), \]

where

1. \( \in_1 \) is the membership relation \( a \in_1 B \) between \( a \in M \) and \( B \subseteq M \)
2. \( \in_2 \) is the membership relation \( B \in_2 \mathcal{C} \) between \( B \subseteq M \) and \( \mathcal{C} \subseteq \mathcal{P}(\mathcal{M}) \)
3. \( (M, \ldots) \) is a first order structure.
A (full) second order structure is a strict 2-sorted structure

$$\mathcal{M} = (\{\mathcal{P}(M), M\}, \in, R, ...),$$

where

1. $\in$ is the membership relation $a \in B$ between $a \in M$ and $B \subseteq M$
2. $R, ...$ are relations, functions and constants between elements of $M \cup \mathcal{P}(M)$. 
The database

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Department</th>
<th>Salary</th>
<th>Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>male</td>
<td>D</td>
<td>2500</td>
<td>Mary</td>
</tr>
<tr>
<td>Mary</td>
<td>female</td>
<td>E</td>
<td>2600</td>
<td>Paul</td>
</tr>
<tr>
<td>Laura</td>
<td>female</td>
<td>B</td>
<td>2500</td>
<td>Mary</td>
</tr>
</tbody>
</table>

is a 4-sorted structure with the sorts

1: Name-sort
2: Gender-sort
3: Department-sort
4: Salary-sort
The database is

\[(\{M_1, M_2, M_3, M_4\}, R)\]

where

\[M_1 = \{\text{John, Mary, Laura, Paul}\}\]
\[M_2 = \{\text{female, male}\}\]
\[M_3 = \{B, D, E\}\]
\[M_4 = \{2500, 2600\}\]

and

\[R \subseteq M_1 \times M_2 \times M_3 \times M_4 \times M_1\]

is the relation indicated by the database.
Example

The two table (relation) relational database

\[ R_1: \begin{array}{|c|c|c|}
\hline
 & \text{John} & \text{Mary} \\
\hline
\text{male} & \text{female} \\
\text{D} & \text{E} \\
\text{B} & \text{B} \\
\text{E} & \text{B} \\
\hline
\end{array} \quad R_2: \begin{array}{|c|c|c|}
\hline
 & A & B \\
\hline
\text{1000} & 1300 \\
\text{1500} & 2500 \\
\text{2600} & 2600 \\
\hline
\end{array} \]

is another example of a 4-sorted structure, this time with two relations.
More about many sorted structures
Definition

A many-sorted structure

\[ M = (\{ M_1, \ldots, M_n \}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l) \]

is isomorphic to

\[ M' = (\{ M'_1, \ldots, M'_n \}, R'_1, \ldots, R'_m, f'_1, \ldots, f'_k, c'_1, \ldots, c'_l) \]

if there is a bijection \( \pi : \bigcup_s M_s \rightarrow \bigcup_s M'_s \) such that

1. \( \pi \restriction M_i : M_i \rightarrow M'_i \) is bijection for \( i = 1, \ldots, n \)
2. \( R_i(a_1, \ldots, a_s) \iff R'_i(\pi(a_1), \ldots, \pi(a_s)) \)
3. \( \pi f_i(a_1, \ldots, a_s) = f'_i(\pi(a_1), \ldots, \pi(a_s)) \)
4. \( \pi c_i = c'_i \)
Definition

A many-sorted structure

\[ \mathcal{M} = (\{M_1, \ldots, M_n\}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l) \]

is an extension of a many-sorted structure

\[ \mathcal{M}' = (\{M'_1, \ldots, M'_n\}, R'_1, \ldots, R'_m, f'_1, \ldots, f'_k, c'_1, \ldots, c'_l) \]

if

1. \( M'_i \subseteq M_i \) for \( i = 1, \ldots, n \)
2. \( R'_i(a_1, \ldots, a_s) \iff R_i(a_1, \ldots, a_s) \) for \( a_1, \ldots, a_s \in \bigcup_{i=1}^n M'_i \)
3. \( f'_i(a_1, \ldots, a_s) = f_i(a_1, \ldots, a_s) \) for \( (a_1, \ldots, a_s) \in \text{dom}(f') \).
4. \( c'_i = c_i \)

Then \( \mathcal{M}' \) is a substructure of \( \mathcal{M} \).
**Definition**

A **Chu transform** between Chu spaces \((A, X, R)\) and \((A', X', R')\) is a pair \(h, g\) of mapping such that

1. \(h : A \rightarrow A'\)
2. \(g : X' \rightarrow X\)
3. \(\forall a \in A \forall x' \in X' (R'(h(a), x') \leftrightarrow R(a, g(x'))))\)
Chu transforms

Example

Transforming (monadic) second order logic into first order logic:

- $A$: sentences of single sorted monadic second order logic
- $X$: single sorted structures $(M, ...)$
- $R$: truth
- $A'$: sentences of 2-sorted first order logic
- $X'$: second order domain $(\mathcal{P}(M), M, \in, ...)$
- $R'$: truth
- $h$: translation of $A$ into $A'$
- $g$: the first order part

where

$$h(\exists x_i \phi) = \exists x_i^0 h(\phi), \quad h(\exists X_i^1 \phi) = \exists x_i^1 h(\phi)$$

$$g((\mathcal{P}(M), M, \in, ...)) = (M, ...)$$
Chu transforms

Example

- Continuous maps in topology:

  - $A$  topological space
  - $X$  its topology
  - $R$  elementhood
  - $A'$ topological space
  - $X'$ its topology
  - $R'$ elementhood
  - $h$  continuous mapping $A \rightarrow A'$
  - $g$  $g(U) = \{ a \in A : f(a) \in U \}$ i.e. preimage

  Now $a \in g(U) \iff h(a) \in U$. 
Example

Important special case: $A \subseteq A', \ X' \subseteq X, \ h = g = id$. 

Definition

A reduct of a many-sorted structure

\[ M = (\{ M_1, \ldots, M_n \}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l) \]

is obtained by leaving out some sorts, relations, functions, and constants, as in

\[ M' = (\{ M_1, \ldots, M_{n-1} \}, R_1, \ldots, R_{m'}, f_1, \ldots, f_{k'}, c_1, \ldots, c_{l'}) \]

Relations, functions, and constants which are not meaningful are at the same time dropped. Respectively then, \( M \) is an expansion of \( M' \). A reduct may also have the same sorts but fewer relations, etc.
Example

Reducts of a vector space \((F, V, +, \cdot, 0)\):

- The scalar field: \(F = (F, +_F, \cdot_F, 0_F, 1_F)\)
- The vector addition group: \((V, +, 0)\)
Example

Reducts of a second order domain \((\mathcal{P}(M), M, \in, \ldots)\):

- The first order part: \((M, \ldots)\)
- The second order part: \((\mathcal{P}(M), M, \in)\)
Example

A reduct of the database

<table>
<thead>
<tr>
<th></th>
<th>John</th>
<th>male</th>
<th>IV</th>
<th>2500</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mary</td>
<td>female</td>
<td>V</td>
<td>2600</td>
<td>Paul</td>
</tr>
<tr>
<td></td>
<td>Laura</td>
<td>female</td>
<td>II</td>
<td>2500</td>
<td>Mary</td>
</tr>
</tbody>
</table>

is not obtained by forgetting (e.g.) the number-sort:

<table>
<thead>
<tr>
<th></th>
<th>John</th>
<th>male</th>
<th>IV</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mary</td>
<td>female</td>
<td>V</td>
<td>Paul</td>
</tr>
<tr>
<td></td>
<td>Laura</td>
<td>female</td>
<td>II</td>
<td>Mary</td>
</tr>
</tbody>
</table>

This is a completely new relation, a so-called projection (not reduct) of the original. In a reduct the entire relation $R$ would drop out.
Example

A reduct of the two table (relation) relational database

\[ R_1 : \begin{array}{cc}
\text{John} & \text{male} \\
\text{Mary} & \text{female} \\
\text{Laura} & \text{female}
\end{array} \quad R_2 : \begin{array}{c}
\text{A} \\
\text{B} \\
\text{C} \\
\text{D} \\
\text{E}
\end{array} \begin{array}{c}
1000 \\
1300 \\
1500 \\
2500 \\
2600
\end{array} \]

obtained by forgetting the number-sort is

\[ R_1 : \begin{array}{cc}
\text{John} & \text{male} \\
\text{Mary} & \text{female} \\
\text{Laura} & \text{female}
\end{array} \]

Relation \( R_2 \) drops out as it uses the abandoned number-sort.
Definition

A single sorted reduct of a many sorted \( L \)-structure

\[
\mathcal{M} = (\{ M_s : s \in \text{sort}(L) \}, R_1, \ldots, R_m, f_1, \ldots, f_k, c_1, \ldots, c_l)
\]

is any reduct

\[
\mathcal{M}' = (\{ M_s \}, R_1, \ldots, R_{m'}, f_1, \ldots, f_{k'}, c_1, \ldots, c_{l'})
\]

to an \( L' = \{ s \} \), where \( s \in \text{sort}(L) \). Relations, functions, and constants which are not meaningful are dropped.

Sometimes the single sorted reducts determine the many sorted structure, but usually not (because the single sorted reducts disregard the interaction between sorts.)
The fundamental translation of a relational many-sorted structure \( \mathcal{M} = (\{M_1, \ldots, M_n\}, R_1, \ldots, R_m, c_1, \ldots, c_l) \) is the single sorted expanded structure

\[
\mathcal{M}^\ast = (M_1 \cup \ldots \cup M_n, M_1, \ldots, M_n, R_1, \ldots, R_m, c_1, \ldots, c_l)
\]

where \( M_1, \ldots, M_n \) are new unary predicates.

From the single sorted \( \mathcal{M}^\ast \) the original \( \mathcal{M} \) can be readily recovered. We have left function symbols out because there is no satisfactory way to extend functions from individual sorts to the union of all sorts. Anyway, functions can be transformed to relations and then there is no problem.
Example

- \(((\mathcal{P}(M), M), \in, \ldots)^* = (\mathcal{P}(M) \cup M, \mathcal{P}(M), M, \in, \ldots)\)
- \(\mathcal{M} \cong \mathcal{M}' \iff \mathcal{M}^* \cong \mathcal{M}'^*\)
- For vector spaces we could define the fundamental translation \((\{F, V\}, +, 0)^* = (F \cup V, F, V, +, 0)\) by letting the value of the functions \(+_F, \cdot_F, +_V\) and \(\cdot_V\) to be 0 whenever they are not canonically defined.
- Note: \(\mathcal{M} \not\cong \mathcal{M}^*\)
Syntax and semantics of many sorted logic
Definition

The vocabulary \( L = \text{voc}(\mathcal{M}) \) of a many-sorted structure (model) \( \mathcal{M} = (\{M_1, \ldots, M_n\}, R_1^\mathcal{M}, \ldots, R_m^\mathcal{M}, F_1^\mathcal{M}, \ldots, F_k^\mathcal{M}, c_1^\mathcal{M}, \ldots, c_l^\mathcal{M}) \)

consists of

1. Sort symbols \( s_1, \ldots, s_n \)
2. Relation symbols \( R_1, \ldots, R_m \)
3. Function symbols \( F_1, \ldots, F_k \)
4. Constant symbols \( c_1, \ldots, c_l \)

We write \( L = \{s_1, \ldots, s_n, R_1, \ldots, R_m, F_1, \ldots, F_k, c_1, \ldots, c_l\} \),

\[
\text{sort}(L) = \{s_1, \ldots, s_n\}, \quad \text{rel}(L) = \{R_1, \ldots, R_m\}, \quad \text{fun}(L) = \{F_1, \ldots, F_k\}, \\
\text{con}(L) = \{c_1, \ldots, c_l\}.
\]
Definition

Each vocabulary $L$ has an \textit{arity-function}

$$a_L : \text{rel}(L) \cup \text{fun}(L) \rightarrow \mathbb{N}$$

which tells the arity of each predicate and function symbol, and a \textit{sort-function} $s_L$:

$$s_L(R) \in \text{sort}(L)^{a_L(R)}, \quad s_L(f) \in \text{sort}(L)^{a_L(f)} \times \text{sort}(L), \quad s_L(c) \in \text{sort}(L).$$

We do not have symbols for abstract relations between elements of arbitrary sorts except identity $=$ in lax structures.
Definition

Suppose $L$ is a many sorted vocabulary. The $L$-variables over many-sorted $L$-structures are denoted:

$$x^s, s \in \text{sort}(L)$$

with the intuition (fulfilled in the semantics below) that $x^s$ ranges over the universe $M_s$ of an $L$-structure

$$\mathcal{M} = (\{M_1, \ldots, M_n\}, R_{1\mathcal{M}}, \ldots, R_{m\mathcal{M}}, F_{1\mathcal{M}}, \ldots, F_{k\mathcal{M}}, c_{1\mathcal{M}}, \ldots, c_{l\mathcal{M}}).$$

No variables ranging over “everything”! No variables ranging over elements of two different sort $s$ (but this can be arranged by having a new sort for the “union”).
Definition

Suppose $L$ is a many sorted vocabulary. The $L$-terms are defined as follows:

1. Constants $c$ in $L$ are $L$-terms and $s(c)$ is already defined.
2. $L$-variables $x^s$ are $L$-terms and $s(x^s) = s$.
3. If $t_1, ..., t_n$ are $L$-terms with $s_i = s(t_i)$ and $f \in L$ with $s(f) = \langle s_1, ..., s_n, s \rangle$, then $f t_1 ... t_n$ (or $f(t_1, ..., t_n)$) is an $L$-term and $s(f t_1 ... t_n) = s$. 
Strict many-sorted logic: we allow $t = t'$ only if $s(t) = s(t')$.

Lax many-sorted logic: we allow $t = t'$ for all terms.