LoCo— A Logic for Configuration Problems

M.Aschinger, C. Drescher, G. Gottlob

2012 Oxford Configuration Workshop

13.01.2012

Knowledge Representation for Technical Product Configuration

LoCo— High-level language for configuration problems:

- Easier to formalize problem
- Easier to change formalization
- Automatic translation to executable formats
- Not suitable for hard problems

Technical Product Configuration — Ingredients

A configuration connects components to meet some requirements

- Components have multi-valued attributes
- Feasible value combinations listed in catalogue
- Connections must not violate constraints
- Number of components needed unknown beforehand: Formalisms like SAT, CSP, ..., need finite bounds

Knowledge Representation — Why Use Logic?

- Well-understood formalism, clear semantics
- Expressive: Boolean connectives, quantifiers

Technical Product Configuration — What Logics Are There?

- Description Logics [4],[1]: Tailored for incomplete information, have tree-model property
- $\exists FO_{\rightarrow,\wedge,+}$[3]: Conditionally select subset of extensional finite component database
- Generative CSP (Logic-version) [2]: Accept infinite configurations

Introducing LoCo

Basic Ideas

- Configuration constraints are axiomatized
- Find a configuration $\tilde{=}$ construct a model
- Axioms implicitly bound the number of components

Components

Components are modelled as predicates:

$$Component(id, \vec{x})$$

Components are identified by unique key *id*:

$$(\forall id_i, id_j, \vec{x}, \vec{y})[\ C(id_i, \vec{x}) \wedge C(id_j, \vec{y}) \wedge id_i = id_j\ ] \Rightarrow \vec{x} = \vec{y}$$

Attributes

Attributes are *sorted* and have *unique names*

Attributes other than *id* have finite domains $\mathcal{V}$:

$$(\forall x)\ AttSort(x) \equiv \bigvee_{V \in \mathcal{V}} x = V$$

Key attributes *id* have potentially infinite domain:

$$(\forall x) CompIDSort(x) \Rightarrow (\exists n) x = CompNo(n)$$

There can be unused component IDs

The Catalogue

Any component used conforms to the catalogue:

$$(\forall id, \vec{x})C(id, \vec{x}) \Rightarrow \phi(\vec{x})$$

with $\phi(\vec{x})$ a quantifier-free formula on comparisons

Component taxonomy can be added (tree-like):

$$(\forall id, \vec{x})C(id, \vec{x}) \equiv \bigotimes SC(id, \vec{x})$$

Counting Quantifiers

LoCo uses counting quantifiers as e.g. $(\exists_l^u x)\ \phi(x)$:

The number of different x s.t. $\phi(x)$ is in the range $[l, u]$

In classical logic:

$$\bigvee_{l \leq n \leq u} \big[(\exists x_1, x_2, \ldots, x_n)[\phi(x_1) \wedge \phi(x_2) \wedge \ldots \wedge \phi(x_n)] \wedge [\bigwedge_{i \neq j} x_i \neq x_j] \wedge [(\forall x)\phi(x) \Rightarrow \bigvee_i x = x_i\ ]\big].$$

Connecting Components

Connections between components C_1, C_2 are binary relation:

$$C_1 2 C_2(id_1, id_2)$$

Don't include $C_2 2 C_1(id_2, id_1)$

Connection Axioms I

Binary connections:

$$(\forall id_1, \vec{x})\ C_1(id_1, \vec{x}) \Rightarrow$$
$$(\exists_l^u id_2)\ C_1 2 C_2(id_1, id_2) \wedge C_2(id_2, \vec{y}) \wedge \phi(id_1, id_2, \vec{x}, \vec{y})$$

- ϕ: quantifier free FO formula on attribute comparisons
- Free variables existentially quantified
- Sometimes need to expand right-hand side
- Good to write both directions

Connection Axioms II

One-to-many connections:

$$(\forall id, \vec{x})\, C(id, \vec{x}) \Rightarrow$$
$$(\exists_l^u id_i)\ [\bigvee_i C2C_i(id, id_i) \wedge C_i(id_i, \vec{y})]$$

Finite Bounds I

Lower bounds of zero can be problematic:

$$(\forall id_1, \vec{x})\ C_1(id_1, \vec{x}) \Rightarrow$$
$$(\exists_0^u id_2)\ C_1 2 C_2(id_1, id_2) \wedge C_2(id_2, \vec{y}) \wedge \phi(id_1, id_2, \vec{x}, \vec{y})$$

$|C_1|$ can be infinite even if $|C_2|$ is bounded

Finite Bounds II

Assume there are only finitely many C_2 and $l_1 > 0$:

$$(\forall id_1, \vec{x})\ C_1(id_1, \vec{x}) \Rightarrow$$
$$(\exists_{l_1}^{u_1} id_2)\ C_1 2 C_2(id_1, id_2) \wedge C_2(id_2, \vec{y}) \wedge \phi(id_1, id_2, \vec{x}, \vec{y})$$

$$(\forall id_2, \vec{x})\ C_2(id_2, \vec{x}) \Rightarrow$$
$$(\exists_{l_2}^{u_2} id_1)\ C_1 2 C_2(id_1, id_2) \wedge C_1(id_1, \vec{y}) \wedge \phi(id_1, id_2, \vec{x}, \vec{y})$$

$\lceil \frac{l_1}{u_2} \cdot |C_1| \rceil \leq |C_2|$ and $|C_1| \geq \lceil \frac{l_2}{u_1} \cdot |C_2| \rceil$

Finite Bounds III

Say a component is

- *input* if it's number is known from the start; and
- *generated* otherwise.

Main Result: A consistent LoCo-axiomatization has only finite models if-and-only-if there is a level mapping s.t.

- *input* components are on level one;
- on level *j* are components connected to components from lower levels via axioms with non-zero lower bounds

Can be decided by graph traversal

Non-Local Constraints

Constraints $\phi(id_1, id_2, \vec{x}, \vec{y})$ in "connection axioms" are local:

$$(\forall id_1, \vec{x})\ C_1(id_1, \vec{x}) \Rightarrow (\exists_I^u id_2)\ C_1 2 C_2(id_1, id_2) \wedge C_2(id_2, \vec{y}) \wedge \phi(id_1, id_2, \vec{x}, \vec{y})$$

Add universally quantified global constraints, e.g.:

$$(\forall id_1, id_2, id_3)\ [\ C_1(id_1) \wedge C_2(id_2) \wedge C_3(id_3) \wedge C_1 2 C_2(id_1, id_2) \wedge C_2 2 C_3(id_2, id_3)\ ] \Rightarrow C_1 2 C_3(id_1, id_3)$$

Knowledge Representation in LoCo

Via domain knowledge:

- Component catalogue
- Division of components into *input*, *generated* and *both*
- Connection axioms
- Non-local constraints (connection generating, candidate key, ...)

and instance knowledge:

- The range of the *input* components
- Component and connection literals

Summary So Far

- Finite number of component types
- Finite attribute ranges:
  Finite number of different component instantiations
- Finitely many components in configurations (iff. a level mapping exists)

Ongoing and Future Work

- Determine complexity of reasoning tasks (at least NP hard)
- Identify tractable islands
- Map to executable formats
- Graphical User Interface


Martin Buchheit, Rüdiger Klein, and Werner Nutt.
Constructive problem solving: A model construction approach towards configuration.
Technical Report TM-95-01, DFKI, 1995.


Gerhard Friedrich and Markus Stumptner.
Consistency-based configuration.
In *Configuration Workshop at AAAI'99*, 1999.


Georg Gottlob, Gianluigi Greco, and Toni Mancini.
Conditional constraint satisfaction: Logical foundations and complexity.
In *IJCAI'07*, 2007.


Deborah L. McGuinness and Jon R. Wright.
Conceptual modelling for configuration: A description logic-based approach.
AI EDAM, 12(4):333–344, 1998.

Many-sorted FO

Sorted variables have different ranges:

$$(\forall n\ \exists s)\mathit{StringNumberPair}(s, n)$$

Reduction to classical FO using unary *Sort* predicates:

$$(\forall n\ \exists s)\mathit{NumberSort}(n) \Rightarrow (\mathit{StringSort}(s) \wedge \mathit{StringNumberPair}(s, n))$$

Can axiomatize sort ranges