Configuration Logic - Modelling Architecture Styles

Anastasia Mavridou, Eduard Baranov, Simon Bliudze, Joseph Sifakis

ICE, 4th of June, 2015

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Outline

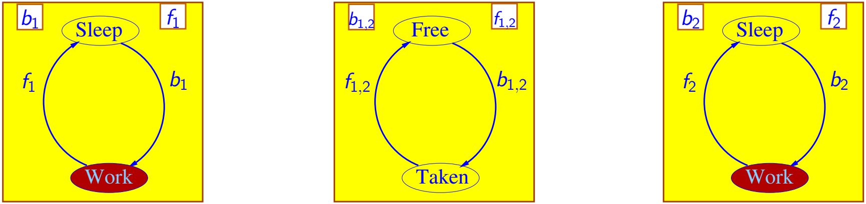
Outline

Reusable design patterns

- Systems are not built from scratch
- Maximal re-use of building blocks
- Maximal re-use of solutions (libraries, design patterns etc.)
- Express coordination constraints in declarative manner

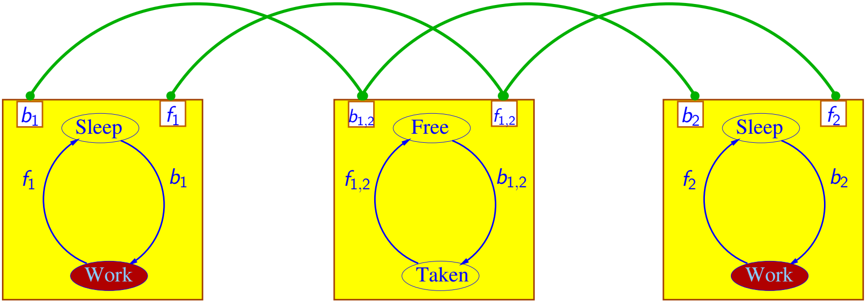
BIP

- Behaviour



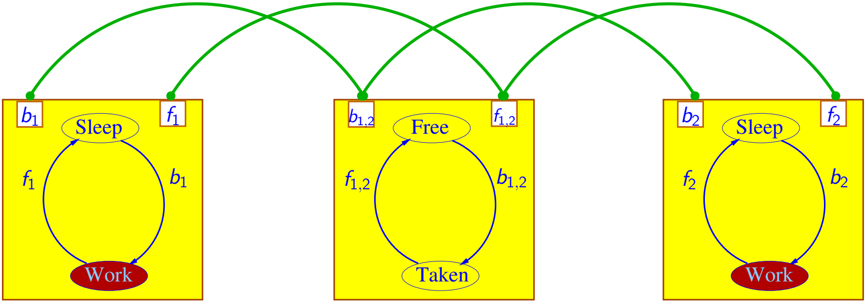
BIP

- Behaviour
- Interaction

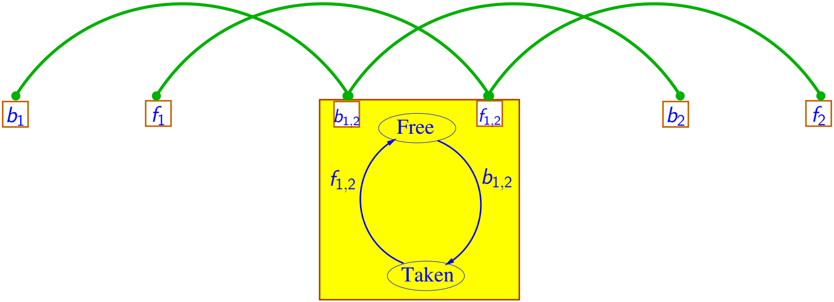


BIP

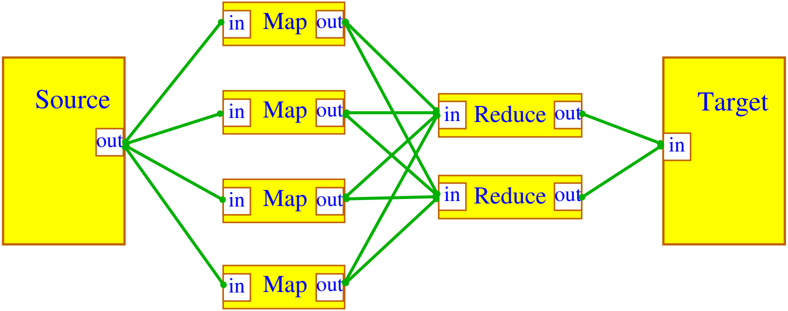
- Behaviour
- Interaction
- Priority



Architecture

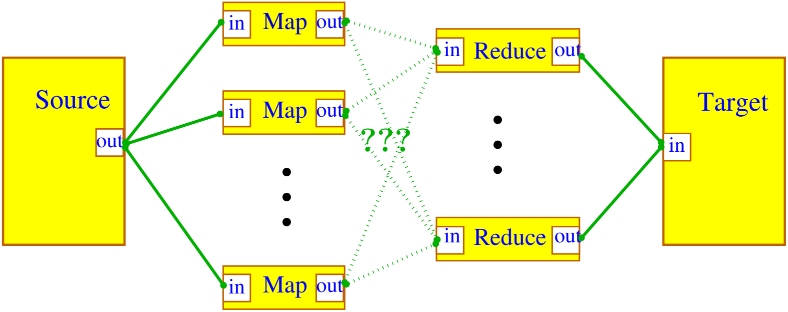


Architecture: Map-Reduce example



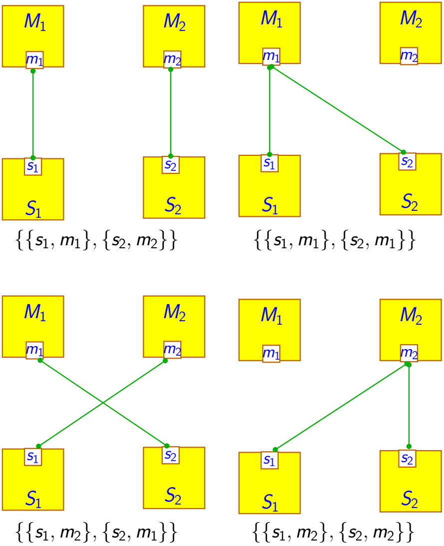
Architecture style: Map-Reduce example

Unknown number of components



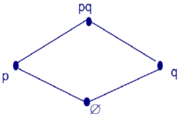
Architecture style 2: Master-Slave example

- Each slave is connected to one master.
- There exist several possible connector allocations

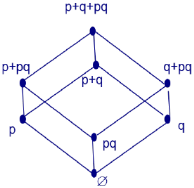


Outline

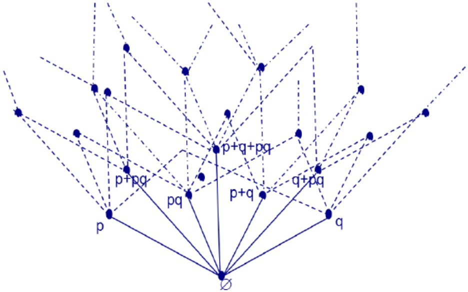
Hierarchy of domains



Interactions $I(P) = 2^P$



Configurations $C(P) = 2^{I(P)\setminus\emptyset}$



Configuration Sets $CS(P) = 2^{C(P)\setminus\emptyset}$

Propositional Interaction Logic (PIL)

Representation of BIP interaction model.

PIL syntax

$$\phi ::= \mathit{true} \mid p \in P \mid \overline{\phi} \mid \phi \vee \phi \mid \phi \wedge \phi$$

A PIL formula defines a configuration

- Each interaction a induces a valuation of ports:
  - If $p \in a$ then $p = \mathit{true}$ else $p = \mathit{false}$
- The formula has to evaluate to true for this valuation

Propositional Configuration Logic (PCL)

PCL is a powerset extension of PIL

PCL syntax

$$f ::= true \mid m \mid f \vee f \mid f \oplus f \mid \neg f \mid f + f$$

The basic elements of the logic are monomials, which are inherited from PIL

PCL semantics

Monomials models any non-empty subset of interactions, which satisfy the PIL version of monomial

For example: $P = \{p, q\}$, a monomial p models $\{\{p\}, \{pq\}, \{p, pq\}\}$
a monomial $p\overline{q}$ models $\{\{p\}\}$

Union: $\gamma \models f_1 \oplus f_2$ if $\gamma \models f_1$ or $\gamma \models f_2$

Complementation: $\gamma \models \neg f$ if $\gamma \not\models f$

PCL semantics 2

Coalescing: $\gamma \models f_1 + f_2$ if there exist non-empty γ_1, γ_2 :
$\gamma = \gamma_1 \cup \gamma_2$, such that $\gamma_1 \models f_1$ and $\gamma_2 \models f_2$

$$p + q$$

$$\{\{p\}, \{pq\}, \{p, pq\}\} \qquad \{\{q\}, \{pq\}, \{q, pq\}\}$$

$$\{\{p, q\}, \{pq\}, \{p, pq\}, \{q, pq\}, \{p, q, pq\}\}$$

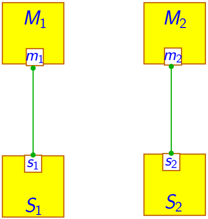
Disjunction: $\gamma \models f_1 \vee f_2$ if either $\gamma \models f_1 \oplus f_2$ or $\gamma \models f_1 + f_2$

Master-Slave example

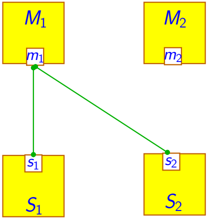
Each slave is connected to one master.

$$(f_{1,1} \oplus f_{1,2}) + (f_{2,1} \oplus f_{2,2})$$

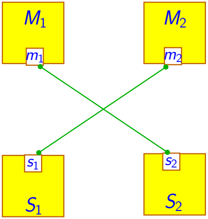
where $f_{i,j} = s_i \wedge m_j \wedge \overline{s_{i'}} \wedge \overline{m_{j'}}$



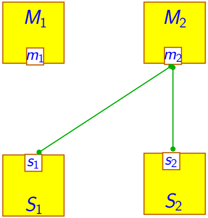
$\{\{s_1, m_1\}, \{s_2, m_2\}\}$



$\{\{s_1, m_1\}, \{s_2, m_1\}\}$



$\{\{s_1, m_2\}, \{s_2, m_1\}\}$



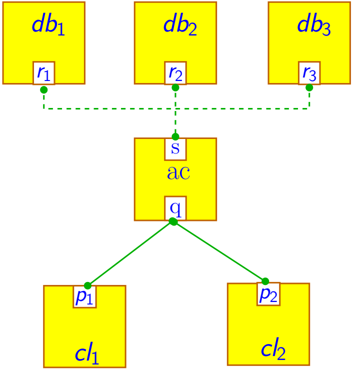
$\{\{s_1, m_2\}, \{s_2, m_2\}\}$

Database access example & Closure operator

Database access example

- Clients can connect to the database only through an access controller
- Connections between databases are not restricted in any way

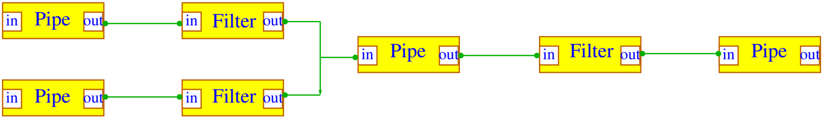
$$\sim(cl1.p1\ ac.q) \wedge \sim(cl2.p2\ ac.q) \wedge$$
$$(\overline{cl1.p1}\ \ \overline{cl2.p2} \vee \overline{db1.r1}\ \ \overline{db2.r2}\ \ \overline{db3.r3})$$



Closure Operator $\sim f = f + true$

- Allows to add anything to a configuration satisfying f.
- Useful for writing specifications

Pipes-Filters example



Two component types: Pipe(P) and Filter(F)

- Each input of a filter is connected to an output of a single pipe

$$\forall f:F.\ \exists p:P.\ \sim(f.in\ p.out) \wedge \forall p':P(p \neq p').\ (\overline{f.in} \vee \overline{p'.out})$$

- Each output of a filter is connected to an input of a single pipe

$$\forall f:F.\ \exists p:P.\ \sim(f.out\ p.in) \wedge \forall p':P(p \neq p').\ (\overline{f.out} \vee \overline{p'.in})$$

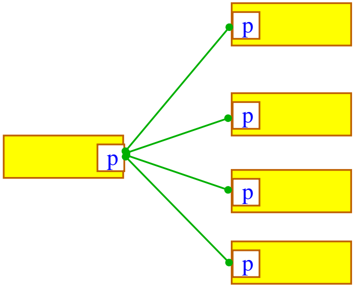
- The output of a pipe is connected to at most one filter

$$\forall p:P.\ \exists f:F.\ \forall f':F(f \neq f').\ (\overline{p.out} \vee \overline{f'.in})$$

Star example

One central component is connected to every other component and no other interactions are allowed

$$\exists s : T.\ \Sigma c : T(c \neq s).\ \sharp(c.p\ s.p)$$

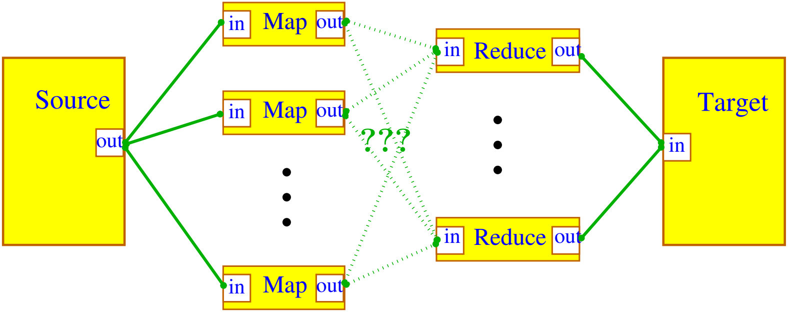


Exact interaction:
$\sharp(c_1.p_1\ c_2.p_2\ \dots\ c_n.p_n)$

- Only ports in the argument participate in the interaction

$$\sharp(c_1.p_1, \dots, c_n.p_n) \stackrel{def}{=} \bigwedge_{i \in [1,n]} c_i.p_i \ \wedge \bigwedge_{i \in [1,n]} \ \bigwedge_{p \in c_i.P \setminus \{p_i\}} \overline{c_i.p} \ \wedge$$
$$\bigwedge_{T \in \mathcal{T}} \left(\forall c : T(c \notin \{c_1, \dots, c_n\}).\ \bigwedge_{p \in c.P} \overline{c.p} \right).$$

Map-Reduce example



$$\sum m : Map.\ \sharp(m.in\ \ s.out) + \sum r : Reduce.\ \sharp(r.out\ \ t.in) +$$
$$\sum m : Map.\ \sum r : Reduce.\ \sharp(m.out\ \ r.in)$$

Normal form

Any configuration set $\{\gamma_1, \ldots, \gamma_n\}$ can be expressed by its characteristic formula:

$$f = \bigoplus_{i=1}^{n} \sum_{a \in \gamma_i} m_a$$

Theorem

Each PCL formula has an equivalent normal form representation

Rewriting rules

$$\frac{g \wedge (f_1 \oplus f_2)}{(g \wedge f_1) \oplus (g \wedge f_2)} \qquad \frac{\neg (f_1 \oplus f_2)}{(\neg f_1) \wedge (\neg f_2)}$$

$$\frac{g + (f_1 \oplus f_2)}{(g + f_1) \oplus (g + f_2)} \qquad \frac{f_1 \vee f_2}{f_1 \oplus f_2 \oplus f_1 + f_2}$$

Rewriting rules 2

$$\frac{\neg \sum_{i\in I} f_i\,, \quad \forall i \in I,\ \overline{f_i} = \bigvee_{j\in J_i} m_j\,, \quad \text{all } f_i \text{ and } m_j \text{ are monomials}}{\bigoplus_{i\in I} \bigvee_{j\in J_i} m_j \oplus \Big(\bigwedge_{i\in I} \bigvee_{j\in J_i} m_j\Big) + true}$$

$\neg\,(p + pq)$

- An interaction is not satisfied by p or $pq \Rightarrow \Big(\overline{p} \wedge (\overline{p} \vee \overline{q}\,)\Big) + true$
- One of the monomials doesn't model any interaction $\Rightarrow \overline{p} \oplus (\overline{p} \vee \overline{q}\,)$

Rewriting rules 3

$$\frac{\sum_{f\in F} f \wedge \sum_{h\in H} h, \quad \text{all } f \in F \text{ and } h \in H \text{ are monomials}}{\bigoplus_{\substack{X \subset F \times H \\ X|_F = F, X|_H = H}} \sum_{(f,h)\in X} f \wedge h}$$

$(p + q) \wedge (r + s)$

- A model can be split into $\gamma_1 \models p$ and $\gamma_2 \models q$ and at the same time $\gamma_3 \models r$ and $\gamma_4 \models s$
- $\gamma_1 \cap \gamma_3 = \gamma_{1,3} \models pr$ and $\gamma = \gamma_{1,3} \cup \gamma_{2,3} \cup \gamma_{1,4} \cup \gamma_{2,4}$
- If all $\gamma_{i,j}$ are non-empty, then $\gamma \models pr + qr + ps + qs$
- If $\gamma_{i,j}$ is empty, the corresponding monomial has to be discarded. All such formulas are combined with the union operator.

Outline

Extensional vs intentional

Extensional

Configurations are built by adding interactions explicitly

Master-Slave example

$$(s_1 m_1 \overline{s_2}\ \overline{m_2} \oplus s_1 m_2 \overline{s_2}\ \overline{m_1})+$$
$$(s_2 m_1 \overline{s_1}\ \overline{m_2} \oplus s_2 m_2 \overline{s_1}\ \overline{m_1})$$

Intentional

Configurations are specified by imposing constraints

Database access example

$$\sim (cl_1.p_1\ ac.q) \wedge \sim (cl_2.p_2\ ac.q) \wedge$$
$$(\overline{cl_1.p_1}\ \overline{cl_2.p_2} \vee \overline{db_1.r_1}\ \overline{db_2.r_2}\ \overline{db_3.r_3})$$

We can combine both!

$$\sum cl : Cl.\ \sharp(cl.p\ ac.q) + \left(\forall cl : Cl.\ \overline{cl.p} \wedge \exists db : Db.\ db.r\right)$$

Conclusion

(Higher-order) PCL allows the specification of architecture styles

- A powerset extension of the Boolean logic on ports
- Is decidable through an algoritmically computable normal form
  - Implemented in Maude
- Combines extensional and intentional approaches
- Can be applied to any component-based formalism, where ports are connected by n-ary connectors.

Future work

- Search for alternative normal forms or sublogics, which allow efficient manipulation
- Graphical notation for architecture styles definition
- Specifically in the BIP context
  - Data transfer
  - Include priorities

Thank you for your attention!