The Joy of Declarative and Concurrent Programming

Jean-Marie Jacquet – CoordiNam Lab

Concurrent logic programming

The framework

$grand\_parent(X,Y) \leftarrow parent(X,Z), parent(Z,Y)$
$parent(jules, marie)$
$parent(marie, antoine)$
$? - grand\_parent(jules, C)$

Issues

- Concurrency (evaluate in parallel)
- Constraints (equality → constraints)
- Contextual programming (handle contexts)

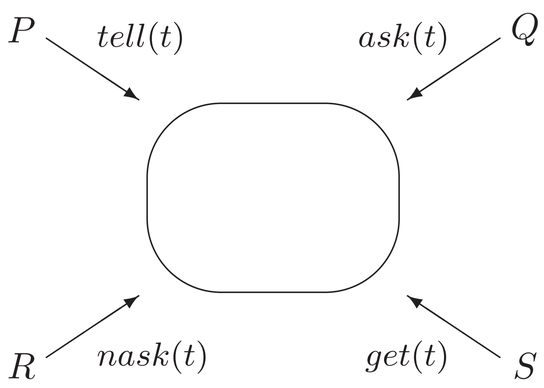
Applications

- Casubel : expert system in estate planning
- Expesurf : expert system in multi-layer surface engineering
- BEM : business event manager

Casubel

Coordination languages and models

BachT, the basic model



$$C ::= tell(t) \mid ask(t) \mid nask(t) \mid get(t)$$
$$A ::= C \mid A\ ;\ A \mid A \parallel A \mid A + A$$

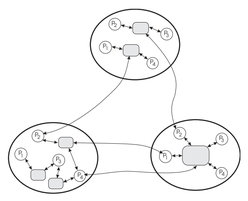
Properties :

- associative memory
- asynchronous communication
- persistent broadcast communication
- decoupling in space and time

$$<tell(t), \sigma> \longrightarrow <E, \sigma \cup \{t\}>$$
$$<ask(t), \sigma \cup \{t\}> \longrightarrow <E, \sigma \cup \{t\}>$$
$$<nask(t), \sigma> \longrightarrow <E, \sigma> \quad \text{for t } \notin \sigma$$
$$<get(t), \sigma \cup \{t\}> \longrightarrow <E, \sigma>$$

Extensions

- structured data and variables
- processes as active data
- time (relative and absolute)
- multiplicity
- chemical abstraction
- distribution
- relations
- mobility



No master manager

Semantics

Declarative

$S \models_{\mathrm{I}} A$ iff $A \in S$
$S \models_{\mathrm{I}} (H \leftarrow \overline{B})$ iff
$S \models_{\mathrm{I}} H$ whenever $S \models_{\mathrm{I}} \overline{B}$

Operational

$$\frac{\mathrm{P} \vdash \mathrm{A}_1\ [\theta_1], \cdots, \mathrm{P} \vdash \mathrm{A}_m\ [\theta_m]}{\mathrm{P} \vdash \mathrm{A}_1,\ldots,\mathrm{A}_m\ [\rho(\theta_1,\ldots,\theta_m)]}$$

$$\frac{\mathrm{P} \vdash \overline{B}\theta\ [\sigma]}{\mathrm{P} \vdash \mathrm{A}\ [\theta\sigma]} \quad \text{if} \begin{cases} (\mathrm{H} \leftarrow \overline{B}) \in \mathrm{P} \\ \theta = \mathrm{mgu(A,H)} \end{cases}$$

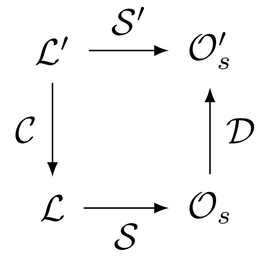
$$\langle A, \epsilon \rangle \longrightarrow \langle \overline{B_1}, \theta_1 \rangle \longrightarrow \cdots$$

Denotational

$$\Psi_{den}(\mathrm{F})(\mathrm{P})((A_1, \cdots, A_m)(\sigma)) = \Psi_{den}(\mathrm{F})(\mathrm{P})(A_1)(\sigma) \ \tilde{\|}\ \cdots \ \tilde{\|}\ \Psi_{den}(\mathrm{F})(\mathrm{P})(A_m)(\sigma)$$
$$\cdots$$
$$\Psi_{\|}(\mathrm{F})(\mathrm{p}_1, \mathrm{p}_2) = \{(\omega, \mathrm{F}(\mathrm{p'}_1, \mathrm{p'}_2)) : \cdots\} \cup \cdots$$

Completely different semantics than for classical concurrency

Expressiveness



$\mathcal{L}(tell)$
$< \mathcal{L}(tell, ask) < \mathcal{L}(tell, get)$
$< \mathcal{L}(tell, ask, get, nask)$

Future work

Foundations

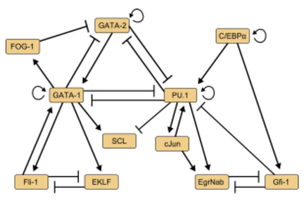
- extensions (e.g. probability, relation and time, capacity)
- programming environments
- implementations
- links with other models (e.g. Reo, mcrl2)
- logics

Blockchain

(image DataNews (21/03/18))

How to coordinate modern distributed systems ?

Bio-algorithmics



(image PLoS One 2011;6(8))

How to apply process algebra techniques ?

Methodology

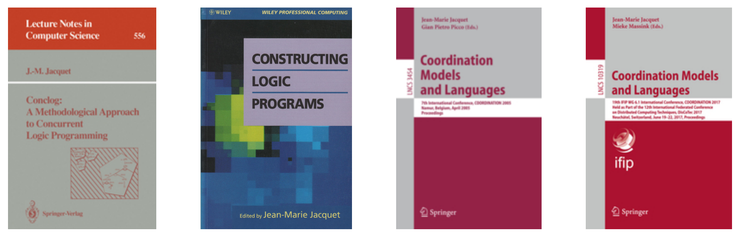
Programming =

Computation + Coordination

- Code each component separately assuming data is available
- Ensure required data is indeed available

Conclusion

- Declarative and concurrent programming is fun and opens interesting research questions
- The CoordiNam Lab offers expertise in functional and logic programming, concurrent programming, formal methods, and coordination languages
- Part of this work has been made in collaboration with A. Brogi, D. Darquennes, K. de Bosschere, I. Linden, L. Monteiro



NADI Namur Digital Institute

Jean-Marie Jacquet
: jean-marie.jacquet@unamur.be
: staff.info.unamur.be/jmj

Université de Namur