A Logic of Behaviour in Context

Chris Banks and Ian Stark

THE UNIVERSITY *of* EDINBURGH
informatics

lfcs | Laboratory for Foundations of Computer Science

MLQA, Edinburgh, September 2012

Introduction

- $\mathcal{LBC}$ – a logic for behaviour in context.
- $\mathcal{LBC}$ is a tractable approach to combining temporal and spatial logics
- with application to continuous biochemical processes.

- $\mathcal{LBC}$ was created as a logic for continuous π-calculus (cπ)
- a continuous time, continuous space process algebra for biochemical systems.

Context introduction

- $\mathcal{LBC}$ introduces the context introduction modality:
  - $C \rhd \phi$

Context introduction

- $\mathcal{LBC}$ introduces the context introduction modality:
  - $C \triangleright \phi$
- Express useful things about biochemical systems.

Context introduction

- $\mathcal{LBC}$ introduces the context introduction modality:
  - $C \triangleright \phi$
- Express useful things about biochemical systems.
- Guarantee operator from Cardelli and Gordon's spatial logic
  - $\phi \triangleright \psi$
    - $P \models \phi \triangleright \psi \iff \forall Q.Q \models \phi \implies P||Q \models \psi$

Context introduction

- $\mathcal{LBC}$ introduces the context introduction modality:
  - $C \triangleright \phi$
- Express useful things about biochemical systems.
- Guarantee operator from Cardelli and Gordon's spatial logic
  - $\phi \triangleright \psi$
    - $P \models \phi \triangleright \psi \iff \forall Q.Q \models \phi \implies P||Q \models \psi$
- Context introduction modality:
  - $Q \triangleright \psi$
    - $P \models Q \triangleright \psi \iff Q||P \models \psi$
- Context introduction is a more tractable interpretation of guarantee.

Logic examples

- $Q \triangleright \mathbf{F}_{[0,3]}([A] \geq c)$
  - If we introduce Q then within 3 time units $[A] \geq c$.

Logic examples

- $Q \triangleright \mathbf{F}_{[0,3]}([A] \geq c)$
  - If we introduce Q then within 3 time units $[A] \geq c$.
- At the top level it might not be very interesting, but

Logic examples

- $Q \triangleright \mathbf{F}_{[0,3]}([A] \geq c)$
  - If we introduce Q then within 3 time units $[A] \geq c$.
- At the top level it might not be very interesting, but
- we can make succinct statements about causality:
- $(Q \triangleright \mathbf{F}([A] \geq c)) \wedge \mathbf{G}([A] < c)$
  - If we introduce Q then eventually $[A] \geq c$, whereas without Q it won't.

Logic examples

- Nesting within temporal modalities gives increased expressive power:
- $\mathbf{F}(Q \triangleright \mathbf{F}([A] < c))$
  - *There exists some point in time* such that if we introduce Q then $[A] < c$.

Logic examples

- Nesting within temporal modalities gives increased expressive power:
- $\mathbf{F}(Q \triangleright \mathbf{F}([A] < c))$
  - *There exists some point in time* such that if we introduce Q then $[A] < c$.
- and bounded modalities allow quite precise quantitative statements:
- $\mathbf{G}_{[3,5]}(Q \triangleright \mathbf{F}_{[0,10]}([A] \geq c))$
  - *For all points in time* from 3 to 5 if we introduce Q then within 10 then $[A] \geq c$.

Logic semantics

For P, a process:

$$P \vDash \phi \mathbf{U}_I \psi \iff \exists t' \in I.\big((P^{t'} \vDash \psi) \wedge (\forall t'' \in I < t'.P^{t''} \vDash \phi)\big)$$

$$P \vDash Q \triangleright \phi \iff (Q \parallel P) \vDash \phi$$

- I is an interval in $[0, \infty)$
- P^t is the process reached from process P after time t.
- Q is a new process.

$$\mathbf{F}_t \phi \equiv \top \mathbf{U}_t \phi$$
$$\mathbf{G}_t \phi \equiv \neg \mathbf{F}_t \neg \phi$$

Model checking

- We build a model using cπ.
- This gives us some ODEs representing the dynamics of the system.
- We can take a discrete numerical simulation trace of the continuous system.
- A numerical ODE solver gives us a *Trace* of the form:

$$(t_0, \vec{c_0}), (t_1, \vec{c_1}), \ldots$$

- A naive model checking algorithm is then easily expressed as a functional program:

Naive recursive model checking

```
check ::  Trace ↦ Formula ↦ Bool

check (t:ts) (atom)          =   valid atom t

check (t:ts) (φ ∧ ψ)         =   ...
check (t:ts) (φ ∨ ψ)         =   ...
check (t:ts) (¬φ)            =   ...

check (t:ts) (φ U_[t0,tn] ψ) =   if (t< t0) then check ts φ U_[t0,tn] ψ
                                 else (t ≤ tn) AND ((check (t:ts) ψ) OR
                                 ((check (t:ts) φ) AND (check (ts) (φ U_[t0,tn] ψ)))

check (t:ts) (Q ▷ φ)         =   check (solve (compose Q (proc (t:ts))))) φ

solve ::  Process ↦ Trace
compose ::  Process ↦ Process ↦ Process
proc ::  Trace ↦ Process
```

Improved model checking

- The naive algorithm has complexity $O(n^f)$ for trace length n and depth f of nested temporal operators.
  - We repeatedly re-compute sub-formulae along the trace,
  - e.g. in $\mathbf{G}(\mathbf{F}\phi)$ if ϕ isn't true until the end of the trace we have cost n^2.
- To improve on this we can make use of the dynamic programming approach used in:

Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.
Machine learning biochemical networks from temporal logic properties.
Transactions on Computational Systems Biology VI, 2006, pp. 68–94.

- where we traverse the trace only once, evaluating all sub-formulae simultaneously.

Hybrid algorithm

- The dynamic programming algorithm avoids unnecessary re-computation.
- However we lose the ability to *short-circuit* the checking, e.g. stop when we find a witness for $\mathbf{F}\phi$.
- By using a hybrid of the two algorithms we can have the best of both.
- The naive algorithm can be restructured as:
  - an outer recursion over the trace
  - and an inner recursion over the sub-formulae.
- Implementation in Haskell exploits any potential laziness and only computes over as much of the trace as we need.

Calls to the solver

- By far the greatest computational cost comes from solving ODEs.
- The key to efficient model checking is reducing the number of calls we have to make to the solver.
- Consider $\mathbf{G}(Q \triangleright \phi)$ – we must compute a new trace introducing Q at every time point.
  - This is *embarrassingly parallel*.
- Context introduction itself doesn't increase complexity, as we can re-write, e.g.:
  - $(Q \triangleright \phi) \wedge (Q \triangleright \psi) \rightarrow Q \triangleright (\phi \wedge \psi)$
  - $Q \triangleright (Q' \triangleright \phi) \rightarrow (Q||Q') \triangleright \phi$
- We have $O(n^d)$ where d is the depth of temporal operators separated by context introductions – $\mathbf{G}(Q \triangleright (\mathbf{G}\phi))$ has $O(n^2)$

Tools

- cπ workbench – implementation in Haskell
  - cπ interactive interpreter
  - species transition system exploration
  - ODE output
  - ODE solvers
    - internal GSL solvers: explicit (RKf45) / implicit (BSimp)
    - external: interface to GNU Octave for LSODE
    - Jacobian computed symbolically by the tool.
  - graph plotting
  - logic and model checker (various algorithms)

Fin

- Acknowledgements:
  - Ian Stark (supervisor)
  - Marek Kwiatkowski (cπ)

- A paper on $\mathcal{LBC}$ is in preparation.
- Continuous π-calculus:

Kwiatkowski, M., & Stark, I.
The continuous π-calculus: A process algebra for biochemical modelling.
In *Proc. Computational Methods in Systems Biology '08*, LNCS 5307, pp.103122. Springer-Verlag, 2008.

Kwiatkowski, M.
A formal computational framework for the study of molecular evolution
Ph.D. Thesis, University of Edinburgh, 2010.

Kwiatkowski, M., & Stark, I.
On Executable Models of Molecular Evolution.
In *Proc. WCSB'11*, pp.105-108, 2011