A new format for process algebras

Vashti Galpin

vashti@cs.wits.ac.za

Programme for Highly Dependable Systems
Department of Computer Science
University of the Witwatersrand

http://www.cs.wits.ac.za/~vashti

Outline and introduction

- process algebras
  - syntax, operational semantics, equivalence semantics
  - examples—CCS, extensions to CCS
- formats
  - existing formats and results
  - new format
  - congruence result
  - comparison results
- fault tolerance and process algebras
  - existing research
  - further research
- conclusions

Process algebras

- concurrency + interaction
- components
  - syntax
  - operational semantics—define labelled transition system, proofs of transitions
  - equivalence semantics—equate processes with same behaviour, bisimulation
- examples
  - CCS
  - CSP
  - ACP
  - extensions to CCS—location, distribution, causality

CCS and its extensions

- syntax
  - $P ::= \text{nil} \mid \alpha.P \mid P + P \mid P|P \mid P\backslash L \mid P[f]$
  - $\alpha \in \{a, b, c, \ldots, \overline{a}, \overline{b}, \overline{c}, \ldots\} \cup \{\tau\}$
  - $L \subset \{a, b, c, \ldots, \overline{a}, \overline{b}, \overline{c}, \ldots\}$
- operational semantics

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

- equivalence semantics, bisimulation—$P \sim Q$ iff for all α
  1. whenever $P \xrightarrow{\alpha} P'$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim Q'$
  2. whenever $Q \xrightarrow{\alpha} Q'$, there exists P' such that $P \xrightarrow{\alpha} P'$ and $P' \sim Q'$

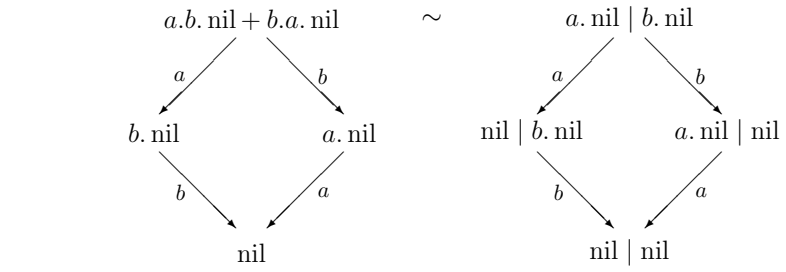
Extensions to CCS

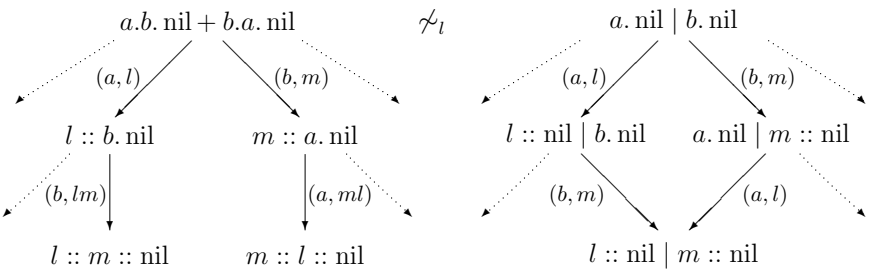
- use additional information to capture characteristics of concurrency
- example—adding location information
  - new syntax: $l :: P$ where $l \in Loc$ disjoint from existing actions
  - new rules for operational semantics, $u \in Loc^*$

$$\frac{}{\alpha.P \xrightarrow{(\alpha,l)} l :: P} \qquad \frac{P \xrightarrow{(\alpha,u)} P'}{P + Q \xrightarrow{(\alpha,u)} P'} \qquad \frac{P \xrightarrow{(\alpha,u)} P'}{P|Q \xrightarrow{(\alpha,u)} P'|Q} \qquad \frac{P \xrightarrow{(\alpha,u)} P'}{l :: P \xrightarrow{(\alpha,lu)} l :: P'}$$

  - new labelled transition system: $\xrightarrow{(\alpha,u)}$
  - new equivalence: bisimulation matches on both action and location
  - example of non-interleaving equivalence

Examples





Formats

- meta-theory of process algebras, deals with rules for operational semantics
- congruence results—a semantic equivalence is a congruence for an operator **op** if

$$\forall 1 \leqslant i \leqslant n,\ \ P_i \sim Q_i \Rightarrow \mathbf{op}(P_1, \ldots, P_n) \sim \mathbf{op}(Q_1, \ldots, Q_n)$$

- number of existing formats—De Simone, GSOS, *tyft/tyxt*, *ntyft/ntyxt*, *panth*
- *tyft/tyxt* format
  - single-sorted signature with standard definition of open terms, closed terms and substitutions, and notion of proof
  - rules have a specific form: y_i's, x_j's and x distinct variables, t_i's and t open terms

$$\frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, \ldots, x_n) \xrightarrow{a} t} \quad \text{or} \quad \frac{\{t_i \xrightarrow{a_i} y_i \mid i \in I\}}{x \xrightarrow{a} t}$$

  - given a signature, a set of rules in *tyft/tyxt* format then bisimulation is a congruence for all operators

A new format

- why?
  - extensions to CCS have structured/non-atomic labels
  - schematic approach no longer works
  - require more general definition of bisimulation—work with equivalences over labels; for example, pomset bisimulation
- extended *tyft/tyxt* format
  - many-sorted signature with distinguished sort for process terms P, plus condition

$$\mathbf{op} : s_1, \ldots : s_n \to s,\ \ s \neq \mathsf{P} \Rightarrow s_i \neq \mathsf{P}\ \ \forall 1 \leqslant i \leqslant n$$

  - terms that have sort other than P can only appear as labels
  - similar notions of open terms, closed terms, substitutions and proofs

– rule format

$$\frac{\{p_i \xrightarrow{\lambda_i} y_i \mid i \in I\}}{f(\eta_1, \ldots, \eta_m, x_1, \ldots, x_n) \xrightarrow{\lambda} p} \quad \text{or} \quad \frac{\{p_i \xrightarrow{\lambda_i} y_i \mid i \in I\}}{x \xrightarrow{\lambda} p}$$

  * y_i's, x_j's and x distinct variables of sort P
  * p_i's and p open terms of sort P
  * η_k's, λ_i's and λ open terms of sort other than P
  * conditions on variables of sort other than P that appear in open terms

- work with more general bisimulation definition
- assume $\equiv$ is a congruence over closed terms with sort other than P, then $P \sim_{\equiv} Q$ iff for all closed terms λ
  1. whenever $P \xrightarrow{\lambda} P'$, there exists Q' and λ' such that $Q \xrightarrow{\lambda'} Q'$, $\lambda \equiv \lambda'$ and $P' \sim_{\equiv} Q'$
  2. whenever $Q \xrightarrow{\lambda} Q'$, there exists P' and λ' such that $P \xrightarrow{\lambda'} P'$, $\lambda \equiv \lambda'$ and $P' \sim_{\equiv} Q'$

Congruence result

- given a many-sorted signature and a set of rules that are well-founded, compatible with $\equiv$, then bisimulation with respect to $\equiv$ is a congruence for all operators

  $\forall 1 \leqslant k \leqslant m,\ \mu_k \equiv \nu_k, \quad \forall 1 \leqslant j \leqslant n,\ u_j \sim_{\equiv} v_j \Rightarrow$

  $\mathbf{op}(\mu_1, \ldots, \mu_m, u_1, \ldots, u_n) \sim_{\equiv} \mathbf{op}(\nu_1, \ldots, \nu_m, v_1, \ldots, v_n)$

- proof sketch
  – define a relation containing the processes under consideration and prove it is a bisimulation
  – for each pair in relation, consider transitions from each process and use induction on the depth of the proof of transitions
  – this involves finding a new substitution to generate a proof that a matching transition exists
  – technical details relate to ensuring that a well-defined substitution can be found

Extensions

- how can two rules sets be joined?
- which new transitions will occur?
- what can be said about the relationship between the two equivalences?
- form sum—$R_0 \oplus R_1$
- existing definitions

  Conservative extension no new transitions are added

  Conservative extension up to bisimulation transitions are added but bisimulation remains the same
- need to take account of equivalence over labels
- need to create new equivalence

Conservative extension up to bisimulation with respect to an equivalence

transitions added

original bisimulation up to original equivalence same as new bisimulation up to new equivalence

Refining extension up to bisimulation with respect to an equivalence

transitions added

new bisimulation up to new equivalence is a subset of original bisimulation up to original equivalence

Abstracting extension up to bisimulation with respect to an equivalence

transitions added

original bisimulation up to original equivalence is a subset of new bisimulation up to new equivalence

- what conditions give the different types of extension?

type-1 sum

- no extended *tyft* rule in R_1 with a function symbol from R_0 in the source of the conclusion has a conclusion label with sort from R_0
- no extended *tyxt* rule in R_1 has a conclusion label with sort from R_0

type-0 sum

- type-1
- no extended *tyft* rule in R_1 has a function symbol in the source of the conclusion from R_0

Lemma

- R_0 pure, label-pure, $R_0 \oplus R_1$ type-1
- if last rule used in the proof of a transition is from R_0 then the transition can be proved using rules from R_0

Abstracting extension theorem

- R_0 pure, label-pure, R_1 well-founded, $R_0 \oplus R_1$ type-0
- $\equiv_0 \oplus \equiv_1$ compatible with $R_0 \oplus R_1$
- proof sketch
  - similar to congruence theorem, but more complex
  - define a relation containing the processes under consideration and prove it is a bisimulation
  - for each pair in relation, consider transitions from each process and use induction on the depth of the proof of transitions
  - this involves finding a new substitution to generate a proof that a matching transition exists
  - technical details relate to ensuring that a well-defined substitution can be found

Refining extension theorem

- R_0 pure, label-pure, $R_0 \oplus R_1$ type-1
- $\equiv_0 \oplus \equiv_1$ conservative with respect to $\equiv_0$
- proof sketch
  - work with the contrapositive and show two terms not equated by the original bisimulation cannot be equated by the new bisimulation
  - use the conservativity of the equivalence, the lemma and type-1 to show that no 'fixing' transitions are added
- conservative extension corollary
- can replace label-pureness condition with safety condition
  - new (non-process) functions cannot have a range with an existing sort

Applications

- using the new format to express process algebras
  - CCS (Milner 1989)
  - CCS with locations (Boudol *et al* 1994)
  - multiprocessor CCS (Krishan 1996)
  - pomset process algebra (Castellani 1988)
- using the new format for comparison of bisimulations
  - pomset bisimulation is a proper subset of n multiprocessor bisimulation ($n > 0$)
  - proof sketch
    - not immediate
    - introduce intermediate process algebra
    - show this is a refining extension of multiprocessor bisimulation using variant of theorem with safety
    - show intermediate process algebra bisimulation and pomset process algebra bisimulation are the same

Fault tolerance and process algebra—an overview

- CCS case studies
- CSP case studies
- trace-based approach (Schepers)
- self-similarity (Weber)
- process algebra for replicated systems (Krishan)
- fault-tolerant bisimulations (Janowski)
-
-
-

Krishnan's research

- CCS-based
- replication operator to model replicated synchronous majority voting
- pre-orders to characterise fault tolerance
- relativised
  - $P \prec_C Q$: Q is no more faulty than P with respect to correctness condition C
- notion of fault injection
- considers omission faults, value faults and addition faults

Janowski's research

- introduces faulty transitions to labelled transition systems

  $$\mapsto = \rightarrow \cup \dashrightarrow$$

- fault-tolerant bisimulation, may bisimulation, $P \Subset Q$ iff for all α
  1. whenever $P \xrightarrow{\alpha} P'$, there exists Q' and s such that $Q \xrightarrow{\alpha} Q'$, $\hat{s} = \hat{\alpha}$ and $P' \Subset Q'$
  2. whenever $Q \overset{\alpha}{\mapsto} Q'$, there exists P' and s such that $P \xrightarrow{\alpha} P'$, $\hat{s} = \hat{\alpha}$ and $P' \Subset Q'$
- fault monotonic theory—if correct for n faults, then correct for $< n$ faults
- conditional fault-tolerance—use finite deterministic automaton to say when faults can occur
- process description language—CCS with recursion
- fault description language—subset of CCS including recursion
- suitable for incremental refinement
- applications—two-phase commit, alternating bit protocol, mutual exclusion, distributed consensus

Further work

- virtual redirector project
- application of extensions of CCS to fault-tolerance
- further theoretical work

Conclusions

- introduction of new format
- more syntactic approach
- proof of congruence result
- proof of extension results
- new format can express CCS and extensions to CCS
- can use to compare bisimulations of different process algebras
- overview of process algebras for fault-tolerance