

Measuring concurrency in CCS

Vashti Galpin

University of the Witwatersrand

1. Introduction
2. The measure m
3. CCS
4. Applying m to CCS
5. Results
6. Further work

1. Introduction

- Investigation of issues relating to concurrency
- Algebraic calculi of processes (eg. CCS) model communication and concurrency
- Analysis of concurrent algorithms
 - Message complexity and time complexity — distributed algorithms
 - Speed-up — parallel algorithms
 - Measures of concurrency — structure of concurrent algorithms
- Further evaluation of measures of concurrency required
- Identification of criteria for evaluation of measures of concurrency

2. CCS

- Defines concurrent agents using a small number of operators

over a set of labels representing actions

- Labels α, β, \dots

– a

– \bar{a} — complement of a

– τ — silent or perfect actions

- Operators

$\mathbf{0}$ Nil, inactive agent

$.$ Prefix $\alpha.\mathbf{0}$

$+$ Summation $\alpha.\mathbf{0} + \beta.\mathbf{0}$

$|$ Composition $\alpha.\mathbf{0} | \beta.\mathbf{0}$

\backslash Restriction $(\alpha.\mathbf{0} | \beta.\mathbf{0}) \backslash \alpha$

$[f]$ Relabelling

- Transition rules — examples

– $a.\mathbf{0} \xrightarrow{a} \mathbf{0}$

– $a.b.\mathbf{0} + c.d.\mathbf{0} \xrightarrow{a} b.\mathbf{0}$

– $a.b.\mathbf{0} + c.d.\mathbf{0} \xrightarrow{c} d.\mathbf{0}$

– $a.\mathbf{0} | \bar{a}.\mathbf{0} \xrightarrow{a} \mathbf{0} | \bar{a}.\mathbf{0}$

– $a.\mathbf{0} | \bar{a}.\mathbf{0} \xrightarrow{a} a.\mathbf{0} | \mathbf{0}$

– $a.\mathbf{0} | \bar{a}.\mathbf{0} \xrightarrow{\tau} \mathbf{0} | \mathbf{0}$

– $(a.\mathbf{0} | \bar{a}.\mathbf{0}) \backslash a \xrightarrow{\tau} (\mathbf{0} | \mathbf{0}) \backslash a$

- Equivalences

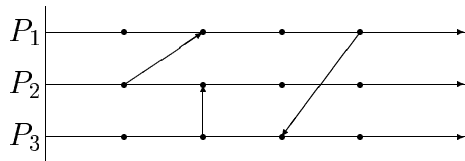
– equating of agents

– differ in treatment of τ actions

3. The measure m

- Model of a distributed system

- internal events
- communication events
- asynchronous, message-passing

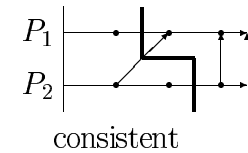
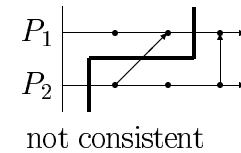


- Happened-before relation

- definition
- partial order on events
- events are concurrent if they are not related

- Consistent cuts

- if event a is in the cut and event b happened before a , then b is in the cut



- vector clocks can be used to check a cut for consistency

- Measure of concurrency

- the less the stopping of one process blocks the other processes the more concurrency the computation exhibits

- tolerance to stopping is related to the ability to be cut consistently

- $m = \frac{\mu - \mu^s}{\mu^c - \mu^s} \in [0, 1]$

Assuming there are q_i events in each process then

$$\mu^s = 1 + q_1 + \dots + q_n$$

$$\mu^c = (1 + q_1)(1 + q_2) \dots (1 + q_n)$$

4. Applying m to CCS

- Identification of differences between message-passing model and CCS

- asynchronous versus synchronous communication

- process creation — $a.(b.\mathbf{0} \mid c.\mathbf{0})$

- difference in expressive power

- Redefinition of model

- new relation \mapsto to describe causality between processes
- new relation \leftrightarrow to describe pairs of actions involved in synchronous communication
- the partial ordering of events is retained with communication events being equated
- cuts, consistent cuts and vector clocks redefined
- can still check for consistency using vector clocks

- Redefinition of measure

- $$m = \frac{\mu - \mu^s}{\mu^c - \mu^s} \in [0, 1]$$

Assuming there are q_i events in each process then

$$\mu^s = (1 + q_1 + \dots + q_n) - c$$

$$\mu^c = \prod_{P_i \in \mathcal{S}} \text{cuts}(P_i) \text{ where}$$

$$\text{cuts}(P_i) = q_i + \prod_{\{P_j | P_i \mapsto P_j\}} \text{cuts}(P_j)$$

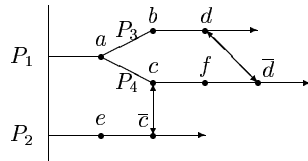
- expressive power — a CCS agent represents a number of computations, each of which can be measured and the measure for the agent can be defined as the ordered pair that consists of the minimum and maximum values found

- Translation from CCS

- Actions are events, and communication between a and \bar{a} form a single event.

- Processes are defined as sequences of Prefixes —
 $a.b.(c.d.\mathbf{0} \mid e.f.\mathbf{0})$

- Example — $(a.(b.d.\mathbf{0} \mid c.f.\bar{d}.\mathbf{0}) \mid e.\bar{c}.\mathbf{0}) \setminus c \setminus d$



- static translation algorithm

- subset of CCS — finite, confluent agents

- Concurrency Measurement Tool

- Use vector clocks to check cuts for consistency

- Expensive — $O(q^n/n^{n-2})$

- Other approaches — antichains, concurrency blocks

- Experiments

- Simple examples

- CCS implementation of two solutions to the Dining Philosophers problem

5. Results

- Evaluation criteria
 - Intuitive understanding of the measure
 - Being well behaved for small examples
 - Compatibility with operators on computations
 - Usability and applicability to actual algorithms
 - Ability to calculate measure for a specific event
 - Expense of computation in terms of both time and space
 - Stability with respect to granularity

- Results of experiments
 - m is not compatible with the operators Prefix and Composition
 - m returns very small values and does not distinguish between algorithms
 - m is expensive to compute
 - New measure defined

$$m_{\text{new}} = \frac{\log \mu - \log \mu^s}{\log \mu^c - \log \mu^s}.$$

- m_{new} better than m , but still problematic

- Conclusions

- The algebraic expression for m is ill-behaved — it is dominated by the value of μ^c
- Modifying the measure can ameliorate this problem
- Successful evaluation of m shows that measurement of concurrency in CCS is feasible
- A methodology for the evaluation of measures of concurrency has been developed

6. Further research

- Development of a better measure based on consistent cuts
- Improvement of the algorithm to count the number of consistent cuts
- Evaluation of other measures of concurrency defined in message-passing formalism