University
of
Glasgow

Generalising Feature Interactions in Email

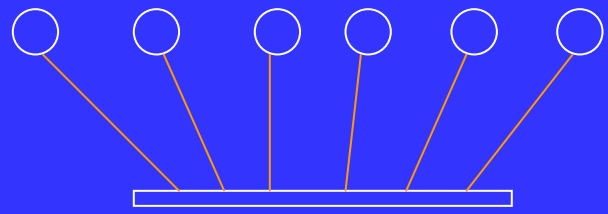
Muffy Calder, Alice Miller
Dept. of Computing Science
University of Glasgow

Motivation

- property based approach to feature interaction analysis
- interaction analysis should
  - *be automated*
  - *generalise to systems containing any number of components*
- based on Hall's email model from FIW'00

Email system

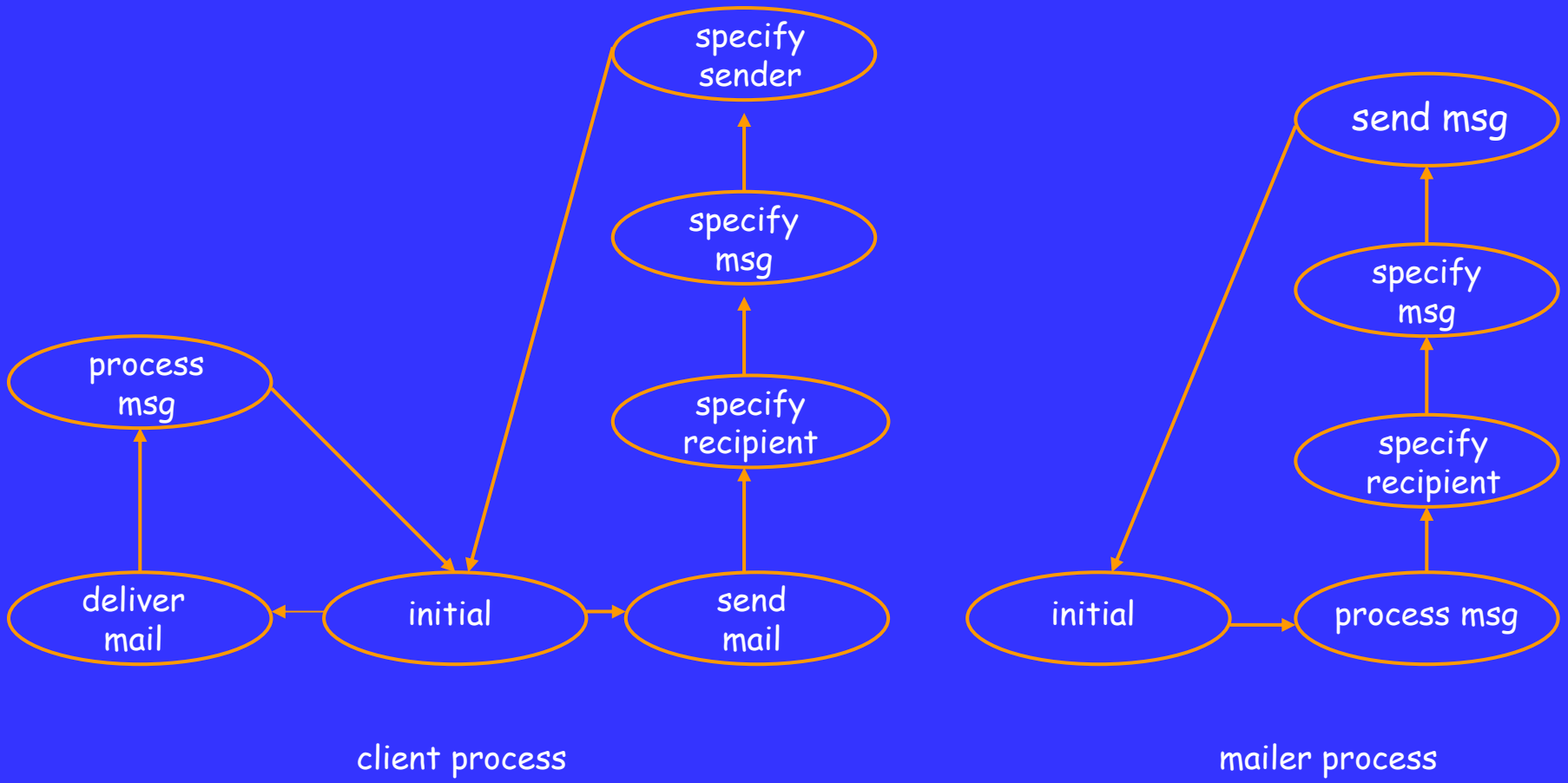
Client server architecture



basic email + features at client *or* server

- encrypt -- key of intended recipient
- decrypt – key of actual recipient
- filter – msgs from address
- forward –msgs to address
- autorespond –to first incoming msgs

Email system



Promela implementation

Clients and the Mailer are processes.

- communication is asynchronous
- channels associated with each client and the mailer
- delivery of mail takes precedence over sending
- busy waiting
- tension between
  - atomicity/number of variables/level of abstraction

Property based approach

Example Properties in linear temporal logic:

- messages are delivered to intended recipients

  $[](p||q)$ where $p = (last\_del\_to_i\_to = i)$

  $q = (last\_del\_to_i\_to = M)$

- messages are forwarded, $client_i$ has forwarding to $client_j$

  $\exists<>(\neg(p||q)$ where $p = (last\_del\_to_j\_to = M)$

  $q = (last\_del\_to_j\_to = j)$

  (i.e. $client_j$ can receive mail not addressed to j)

observation variables: $last\_del\_to_i\_to$, $last\_del\_to_i\_body$, $last\_sent\_from_i\_to$

Property based approach

Feature interaction analysis based on:

$f_0 \,||\, System \models \phi$ but $f_0 \,||\, f_1 \,||\, System \not\models \phi$

for example:

$Client_0 \,||\, Client_1 \,||\, Client_2 \,||\, Client_3 \,||\, Mailer \models \phi$

but

$Client'_0 \,||\, Client_1 \,||\, Client_2 \,||\, Client_3 \,||\, Mailer \not\models \phi$

Reasoning

- Use model-checker (SPIN) for reasoning
- Results take the form

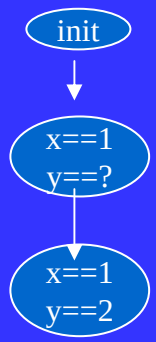
$$\mathcal{M}(p_0 \,||\, p_1 \,||\, p_2 \,||\, p_3 \,||\, \text{mailer}) \models \phi(0,1,\ldots,t)$$

where

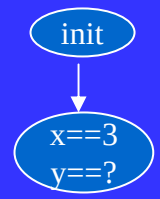
- $p_0 \ldots p_3$ are instances of a *parameterised* process p
- $p_0 \ldots p_3$ are not, in general, isomorphic
- ϕ is temporal logic formula containing free variables indexed by $0,1,\ldots,t \quad 0 \leq t < 3$
- M(….) is the model (Kripke structure) of the concurrent processes

model checking

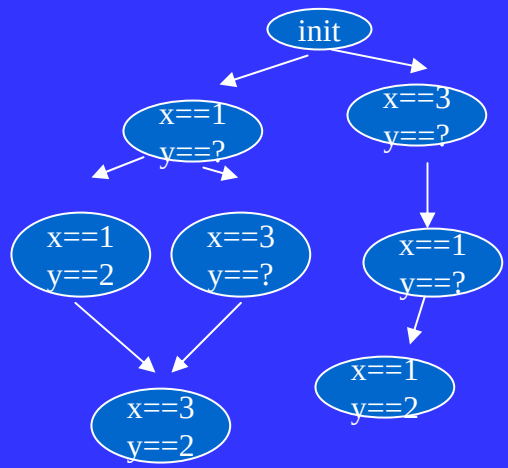
proctype A()
{x=1;y=2}



proctype B()
{x=3}



program:
byte x,y;
init {run A(); run B()}

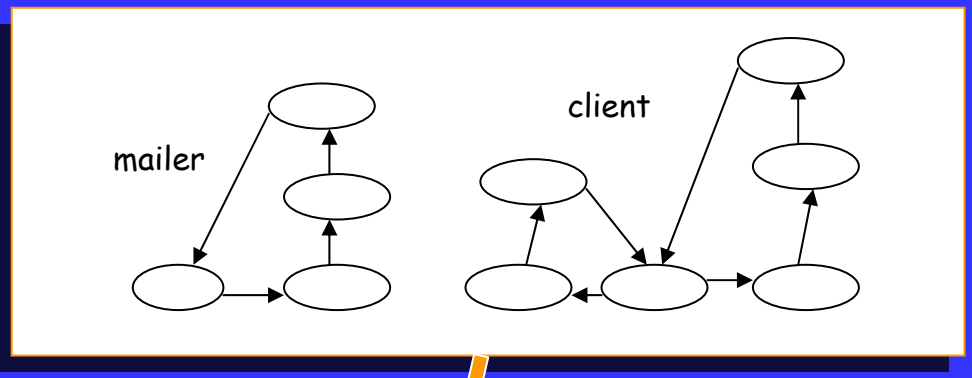


A program/system is the asynchronous interleaving product of the automata.

Reason about satisfaction of LTL formula by considering synchronous product of system with never claim.

overall approach

Mailer and client basic behaviour



9 features

Promela code

LTL

"client$_i$ has filter from client$_j$"

[](¬(last_del_to$_i$ _from==j)

property

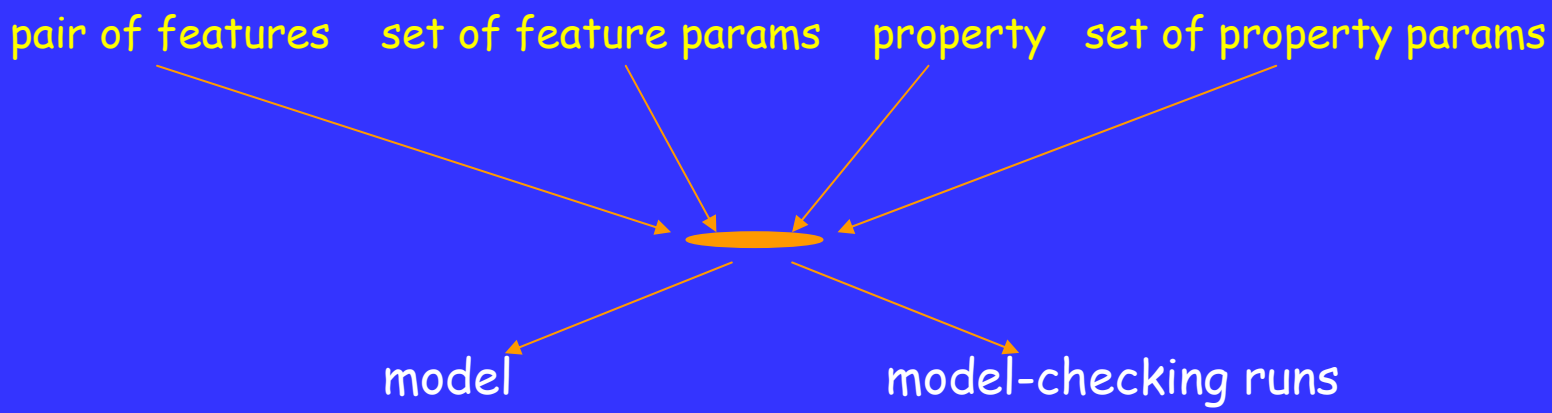
Configuration of features

parameters

SPIN

Answer: 0 errors or counter-example

automation



up to 5 client processes required for this feature set
111 feasible parameter sets, after symmetry reduction

via perl scripts

interactions

- the usual!
- both single user and multiple user
- multiple user agree with Hall results

The interesting question is.. do results scale?

- do they hold for 7 processes, 8 processes, … ?
- for every problem, we eventually run out of memory (or patience)
  - usually around 6 processes

Generalisation

- What we really want is to show is

$$\forall n.\ \mathcal{M}(p_0 \mid\mid p_1 \mid\mid p_2 \mid\mid \ldots \mid\mid p_{n-1} \mid\mid \text{mailer}) \models \phi(0,1,\ldots,t)$$

Not possible with model checking

Undecidable (Apt + Kozen, 1986)

- Induction is very hard
- Is abstraction possible?

Abstraction

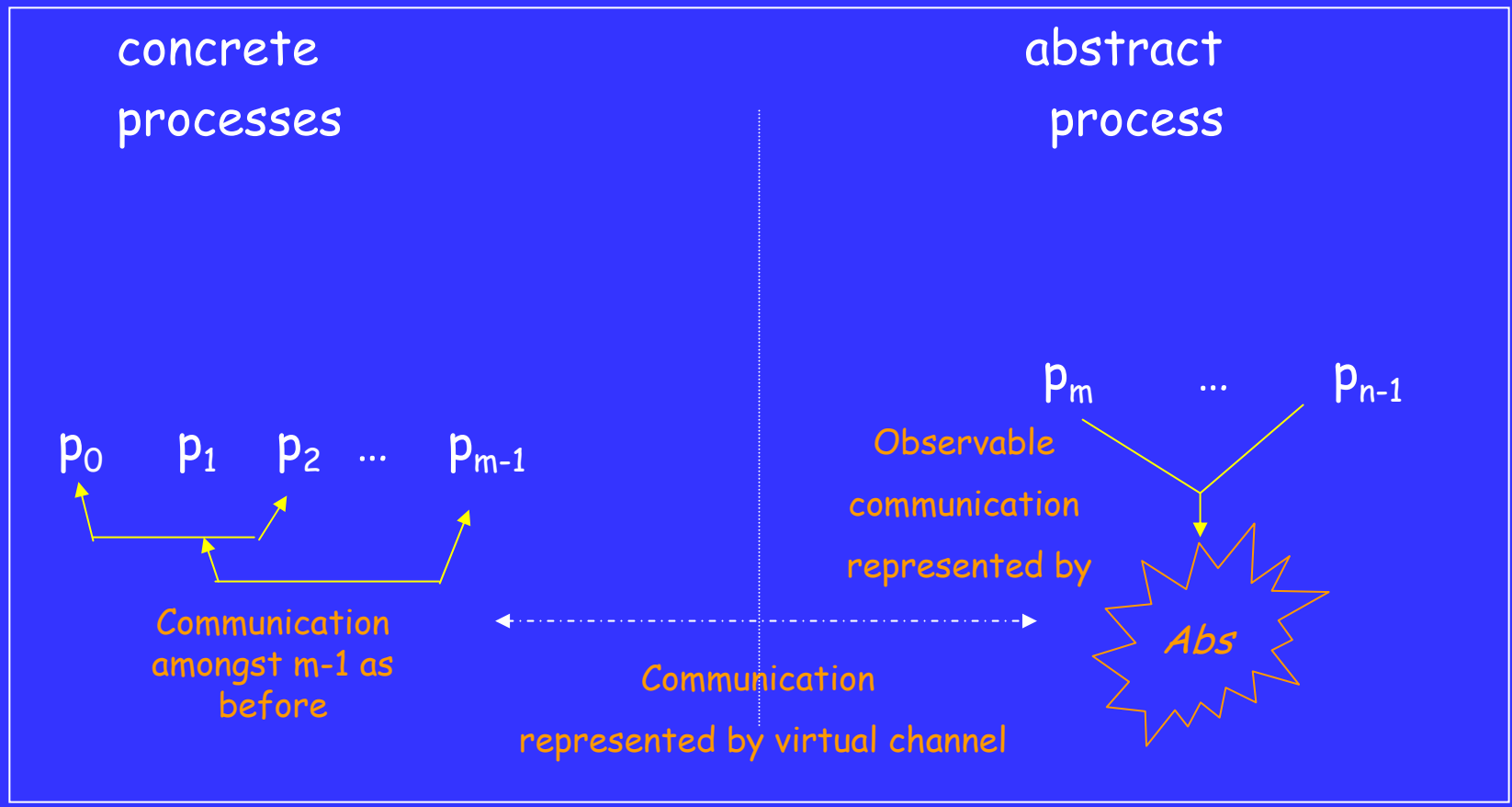
- What do we mean by abstraction?

  - not an abstraction of one system, but of a family of systems

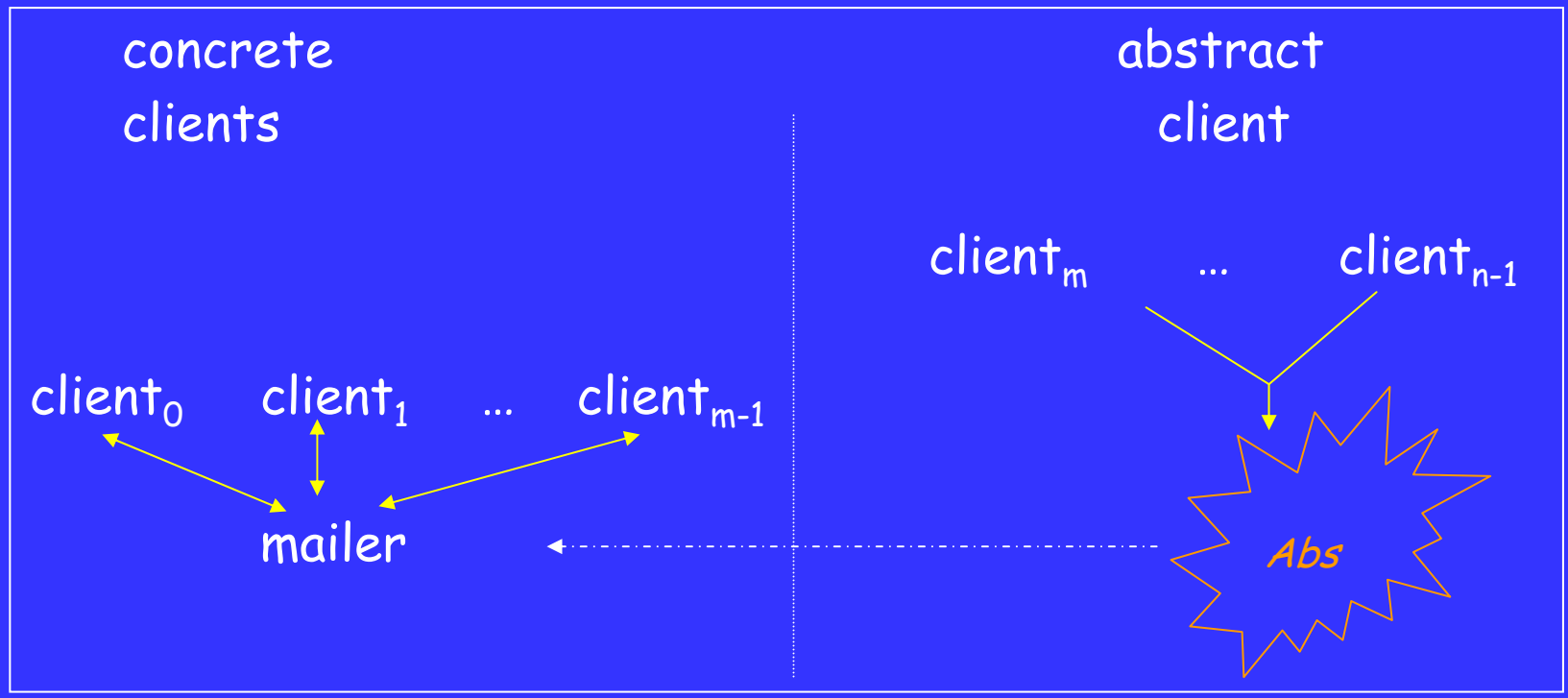
  - Choose appropriate constant *m* such that $t \leq m-1$
    - $p_0, p_1 .. p_{m-1}$ are *concrete*
    - $p_m, p_{m+1} .. p_{n-1}$ are *abstract*
  - Represent the most *general, observable* behaviour of the abstract processes $p_m \,||\, p_{m+1} \,||\ldots||\, p_{n-1}$ by a process *Abs*
  - Modify the interaction between concrete and abstract processes, i.e. modify the concrete processes.

All processes should be generated automatically from p.

Abstract approach



Abstract approach: email



- Only "read" behaviour from *Abs*. (Choice rather than actual read)
- mailer "writes" to *Abs* if not blocked. (Choice).
- Not strictly a conservative extension. *Abs* can send mail if there is mail to to be delivered. Does not affect functional behaviour.

Results

Checking done with perl scripts.

No new interactions (not surprising, given feature set).

Complexity lies between that of m and m+1 (concrete) clients.

m depends on feature parameters and property parameters (essentially union).

Soundness

The p_m, p_{m+1}, … ,p_{n-1} need not be isomorphic, or even observationally equivalent, but we make some assumptions about both concrete and abstract processes:

1. All interaction is through communication channels.
2. All processes are *open symmetric* – behave the same with respect to isomorphic processes - no integer literals or constants in boolean conditions
   - g?x; x==9 => goto label NO
   - g?x; x==var_i => goto label YES

Theorem

$\mathcal{M}(\text{client}_0 \,||\, \text{client}_1 \,||\, \ldots \,||\, \text{client}_{m-1} \,||\, \text{mailer}' \,||\, \mathit{Abs}) \models \phi(0,1,\ldots,t)$

=>

$\forall n.\ \mathcal{M}(\text{client}_0 \,||\, \text{client}_1 \,||\, \ldots \,||\, p_{n-1} \,||\, \text{mailer}) \models \phi(0,1,\ldots,t).$

Proof

Show *simulation*. Depends on way we construct *Abs* and mailer': consider how to match

- concrete process reads from abstract channel
- concrete process write to abstract channel

...

- abstract process reads from concrete channel
- abstract process writes to abstract channel

Conclusions

- property based approach to interaction analysis
  - automated using perl scripts – to tailor model and generate runs.
- abstraction to generalise results about infinite *families* of communicating processes
  - processes are not isomorphic.
- generation of abstract model is straightforward
  - implemented in perl scripts
  - lower bound for m
  - for this feature set abstraction approach is tractable.

further work – is the *abstraction approach* constructing an *invariant?*