Behavioural Equivalences for Distributed Agents

Matthew Hennessy

University of Sussex

Work in Progress

Joint work with Massimo Merro and Julian Rathke

Partially funded by Mikado GC-project and Royal Society

Behavioural Equivalences for Distributed Agents

- Dpi: a language for distributed agents
  - syntax and reduction semantics
  - dynamic capability types for controlling resources
- Behavioural Equivalences
  - dependent on users knowledge
  - contextual versus bisimulation equivalences
- Controlling Mobility
  - using types to control access to sites
  - effect on behavioural equivalences

The Computational Model underlying DPI

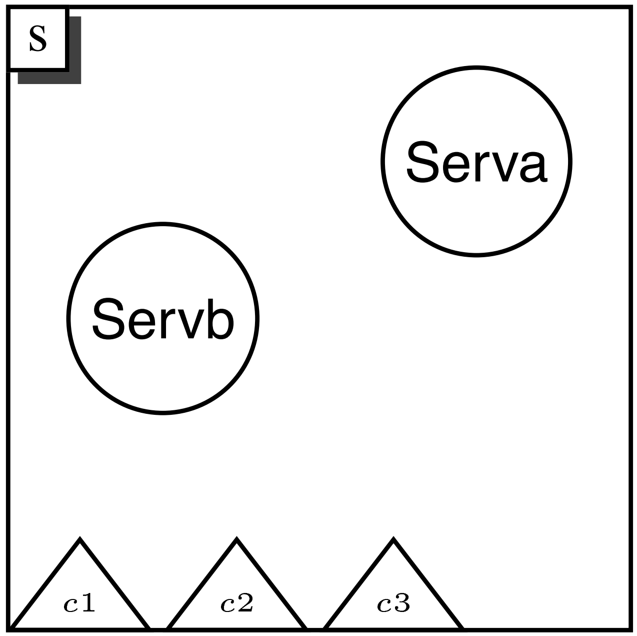
locations/sites: where computations occur. Flat structure; may be generated dynamically.
In DPI all communication is local.

mobile agents: perform computations; move from site to site.
Described by augmented π-calculus processes.

resources: (communication channels) available at specific locations.
In DPI names of resources only have local significance.

types: guarantee controlled access to resources.
Agents may only use resources in accordance with capabilities/permissions they have acquired.
May be generated dynamically.

DPI Processes



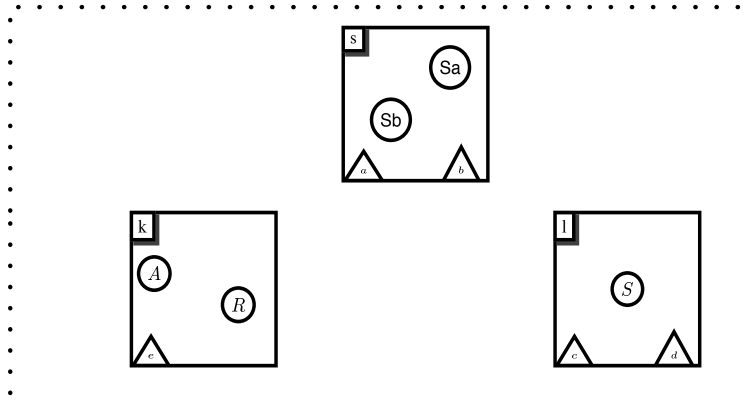
S - site name

$c1, c2, c3$ -resources available at S

$\mathsf{Serva}, \mathsf{Servb}$ - anonymous threads/agents

Agents may move autonomously from site to site

Distributed Systems



A collection of independent distributed *sites* offering **services/resources** to migrating **agents**.

Resources: Modelled by picalculus channels

Agents: Modelled by (augmented) picalculus processes

DPI– a π-calculus with explicit locations

A **System** is a collection of autonomous **threads/agents** each running at explicit locations

Syntax of Systems M-N:

- $l[\![P]\!]$ - the thread P running at site l
- $M \mid N$ - threads running in parallel
- $(\mathsf{new}\, e : \mathrm{T})\ M$ - sharing (typed) information

Example Systems:

$$l[\![P]\!] \mid (\mathsf{new}\, a : \mathrm{A}_{@}k)\ (l[\![Q]\!] \mid k[\![R]\!])$$
$$l[\![d?(x_{@}z)\, P]\!] \mid k[\![(\mathsf{newc}\, a : \mathrm{A})\ \mathsf{goto}\, l.d!\langle a_{@}k\rangle]\!]$$

Threads/Agents in DPI

Syntax of threads, $P - R$:

- $u?(X : \mathrm{T})\, Q$ - *local* input on channel u
- $u!\langle V\rangle\, Q$ - *local* output on channel u
- $\mathsf{goto}\, u.P$ - code movement to site u
- $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$ - testing of names
- $(\mathsf{newc}\, a : \mathrm{A})\ \ P$ - generation of new local channel
- $(\mathsf{newloc}\, k : \mathrm{K})\ \ \mathsf{with}\ P$ - generation of new location with initial code P
- $(\mathsf{newreg}\, n : \mathsf{rc}\langle \mathrm{A}\rangle)\ \ P$ - registration of new channel name to used consistently at multiple locations
- $P \mid Q$ - concurrent code
- $*P$ - iteration

Reduction Semantics example rules

$$k[\![c!\langle V\rangle\, Q]\!] \mid k[\![c?(X : \mathrm{T})\, P]\!] \to k[\![Q]\!] \mid k[\![P\{\!|{}^{V}\!/\!_{X}|\!\}]\!]$$

$$k[\![P \mid Q]\!] \to k[\![P]\!] \mid k[\![Q]\!]$$

$$k[\![\mathsf{goto}\, l.P]\!] \to l[\![P]\!]$$

$$k[\![(\mathsf{newloc}\, l : \mathrm{L})\ \ \mathsf{with}\ C]\!] \to (\mathsf{new}\, l : \mathrm{L})\ \ l[\![C]\!]$$

$$k[\![(\mathsf{newc}\, n : \mathrm{A})\ \ P]\!] \to (\mathsf{new}\, n : \mathrm{A}_{@}k)\ \ k[\![P]\!]$$

with a structural congruence to make it all work

Types via examples

Location type: record of resources known to be available

server site s with type $\mathrm{L}_s = \mathsf{loc}[\mathit{quest} : \mathrm{T}_q,\ \mathit{ping} : \mathrm{T}_p,\ \mathit{kill} : \mathrm{T}_k]$

$$\begin{aligned} s[\![\ \ \mathit{internals} \mid {*}\mathit{quest}?(X : \mathrm{U}_q) \ldots \\ {*}\,\mathit{ping}?(X : \mathrm{U}_p) \ldots \\ {*}\,\mathit{kill}?(X : \mathrm{U}_k) \ldots \quad]\!] \end{aligned}$$

Server s may also be known at *supertypes* of L_s:

$$\mathsf{loc}[\mathit{quest} : \mathrm{T}_q,\ \mathit{ping} : \mathrm{T}_p]$$

$$\mathsf{loc}[\mathit{quest} : \mathrm{T}_q]$$

$$\mathsf{loc}$$

Remote channel types

Server:

receives *data* - x, *return* channel - y, at some *unknown* location - z:

$$s[\![\ldots \mid *quest?(x, y@z)$$
$$\mathsf{goto}\, z.y!\langle isprime(x)\rangle]\!]$$

Type of service at port *quest* - $\mathrm{T}_q = \mathsf{r}\langle \mathrm{U}_q \rangle$, where

$$\mathrm{U}_q = \langle\ \mathbf{int},\ \mathsf{w}\langle \mathbf{bool}\rangle @\mathsf{loc}\ \rangle$$

Client:

$$c[\![(\mathsf{newc}\, r : \mathsf{rw}\langle \mathbf{bool}\rangle)\ \mathsf{goto}\, s.quest!\langle v, r@c\rangle\, \mathsf{stop} \mid r?(z)\ldots]\!]$$

Dynamic types

Replies must come to *me*:

$$l[\![\ \mathit{setup}?(z)\ \ (\mathsf{newloc}\, s : \mathrm{L}_s^z)\ \ \mathsf{with}\ \ {*}\mathit{quest}?(x,y)\ \ \ldots]\!]$$

where

$$\mathrm{L}_s^z = \mathsf{loc}[\mathit{quest} : \mathsf{rw}\langle \mathbf{int}, \mathsf{w}\langle \mathbf{bool}\rangle_{@} z\rangle,\ \mathit{ping} :\ \ \ldots\ ]$$

Acknowledgments will only be sent to location z, instantiated on input via *setup*.

Dynamic type $\mathsf{rw}\langle \mathbf{bool}\rangle_{@} z$ only instantiated at run-time.

Client:

$$\mathit{me}[\![\mathsf{goto}\, l.\mathit{setup}!\langle \mathit{me}\rangle \ldots\ \ ]\!]$$

receives personalised treatment

Shared Interfaces via Registered names

Registered names may be declared at varying locations

Remote bank account server:

$$(\mathsf{newreg}\ \ put : \mathsf{rc}\langle \mathrm{T}_p \rangle,\ get : \mathsf{rc}\langle \mathrm{T}_g \rangle)$$

$$\begin{aligned} \mathit{Server} \Leftarrow s[\![& \mathit{request}?(x : \mathbf{int}, y@z) \\ & \qquad (\mathsf{newloc}\, b : \mathrm{L}_b)\ \mathsf{with} \ldots put,\ get \\ & \qquad \mid \mathsf{goto}\, z.y!\langle b \rangle]\!] \end{aligned}$$

Declared type of new account L_b:

$$\mathsf{loc}[put : \mathrm{T}'_p,\ get : \mathrm{T}'_g]\ \text{ subtypes of } \mathrm{T}_p, \mathrm{T}_g$$

$$\mathit{Client} \Leftarrow me[\![(\mathsf{newc}\, r)\ \mathsf{goto}\, s.\mathit{request}!\langle r@me \rangle\ \mid r?(x) \ldots]\!]$$

Client: - receives name of new bank account

Shared Interfaces

An alternative **bank account** server:

Client generates new location using interface obtained from server

$$Server \Leftarrow (\mathsf{newreg}\ \ put : \mathsf{rc}\langle \mathrm{T}_p\rangle,\ get : \mathsf{rc}\langle \mathrm{T}_g\rangle)$$
$$s[\![request?(y@z)$$
$$\mathsf{goto}\, z.y!\langle put, get\rangle]\!]$$
$$Client \Leftarrow me[\![(\mathsf{newc}\, r)\ \ \mathsf{goto}\, s.request!\langle r@me\rangle \ \ |$$
$$r?(y, z)\, (\mathsf{newloc}\, b : \mathrm{L}^{y,z})\ \ \mathsf{with}\ \ldots code \ldots]\!]$$

Client-side locations must have declared dynamic type

$$\mathrm{L}^{y,z} = \mathsf{loc}[y : \mathrm{T}'_g, z : \mathrm{T}'_p]$$

Type inference: $\Gamma \vdash M$

System M is well-typed relative to the type environment Γ

Environments Γ: $u_1 : \mathbf{base}, \ldots u_2 : \mathsf{loc}, \ldots u_3 : \mathsf{rc}\langle \mathrm{A} \rangle, \ldots, u_4 : \mathrm{A}_{@}w$

Separate judgements required for

- well-formed **environments**: $\Gamma \vdash \mathbf{env}$
- well-formed **types** - requires **subtyping**
- typing **agents**: $\Gamma \vdash_k P$
- typing **identifiers**: $\Gamma \vdash u : \mathrm{T}$

Behavioural equivalence between Systems

- Distinguishing between systems depends on knowledge of locations and their resources

- This knowledge evolves as the systems are used

$$\mathcal{I} \triangleright M \cong_{cxt} N \quad \text{means} \text{ {\scriptsize informally}}$$

M and N have the same behaviour for any user which has the knowledge $\mathcal{I}$ of the resources in M and N.
$\mathcal{I}$ is a type environment

Example:

$$\mathcal{I}, a : \mathsf{r}\langle \mathrm{T} \rangle_{@} k \triangleright \quad k[\![P \mid a?(x)\, Q]\!] \cong_{cxt} k[\![P]\!]$$

if a not in P user can not send on a at k

Effect of type inference

Transmitting the same data v twice:

$$k[\![o_1!\langle v\rangle \,.\, o_2!\langle v\rangle \,.\mathrm{in}?(x) \,. \quad \mathsf{stop}]\!] \quad \stackrel{?}{\cong}_{cxt} \quad k[\![o_1!\langle v\rangle \,.\, o_2!\langle v\rangle \,.\mathrm{in}?(x) \,. \quad \mathsf{if}\ x = v\ \mathsf{then}\ \ \mathrm{yeah}!\langle\rangle]\!]$$

Successful probe:

$$t[\![\mathsf{goto}\, k.o_1?(x) \,.\, o_2?(y) \,. \quad \mathsf{if}\ x = y\ \mathsf{then}\ \mathrm{in}!\langle x\rangle \,.\, \mathrm{yeah}?() \,.\mathsf{goto}\, t.\mathrm{Eureka}]\!]$$

Question: Can probe always be typed ?

Answer: Depends on types of channels: $o_1,\ o_2,\ \mathrm{in}$

Typing the probe

$$\text{In } \mathcal{I}: \quad \begin{array}{rcl} o_1 & \mapsto & \mathsf{rw}\langle \mathsf{r}\langle\rangle\rangle_{@}k \\ o_2 & \mapsto & \mathsf{rw}\langle \mathsf{w}\langle\rangle\rangle_{@}k \\ \text{in} & \mapsto & \mathsf{rw}\langle \mathsf{rw}\langle\rangle\rangle_{@}k \end{array}$$

Probe **not** typeable with standard rule: $\dfrac{\begin{array}{l}\mathcal{I} \vdash x : \mathrm{T}, \;\; y : \mathrm{T}, \\ \mathcal{I} \vdash_k P, \;\; Q\end{array}}{\mathcal{I} \vdash_k \mathsf{if}\ x = y\ \mathsf{then}\ P\ \mathsf{else}\ Q}$

Types at which v is received must be conflated to satisfy type constraint on channel in:

$$\dfrac{\begin{array}{l}\mathcal{I} \vdash x : \mathrm{T}, \; y : \mathrm{U}, \\ \mathcal{I} \vdash_k Q \\ \mathcal{I} \sqcap \langle x : \mathrm{U}\rangle_{@}k \sqcap \langle y : \mathrm{T}\rangle_{@}k \vdash_k P\end{array}}{\mathcal{I} \vdash_k \mathsf{if}\ x = y\ \mathsf{then}\ P\ \mathsf{else}\ Q}$$

Typed contextual equivalence $\mathcal{I} \triangleright M \cong_{cxt} N$

touchstone equivalence between systems

The largest (paramaterised) equivalence which is

- reduction closed - “preserves” reduction relation $M \ \rightarrow\ M'$
- preserves “observations” - a at k can be observed in M if and only if it can also be observed in N
- preserved by $\mathcal{I}$-contexts: If $\mathcal{I} \triangleright M \cong_{cxt} N$ then
  - $\mathcal{I} \vdash O$ implies $\mathcal{I} \triangleright M \mid O \cong_{cxt} Q \mid O$
  - $\mathcal{I}, n : \mathrm{T} \triangleright M \cong_{cxt} N$ implies
    $\mathcal{I} \triangleright (\mathsf{new}\, n : \mathrm{T})\ M \cong_{cxt} (\mathsf{new}\, n : \mathrm{T})\ N$
  - $\mathcal{I}, \mathcal{I}' \triangleright M \cong_{cxt} N$ -preserved by adding new names

Result: Can be characterised using bisimulations

Typed Bisimulations

Need typed actions: $\boxed{\mathcal{I} \triangleright M \xrightarrow{\mu} \mathcal{I}' \triangleright M'}$

Note: Type environment may change

Typed bisimulation equivalence: Largest family of relations satisfying:

If $\mathcal{I} \triangleright M \approx_{bis} N$ then

$$\begin{tikzcd}
\mathcal{I} \triangleright M \arrow[d, "\mu"'] & \text{implies} & \mathcal{I} \triangleright N \arrow[d, Rightarrow, "\hat{\mu}"'] \\
\mathcal{I}' \triangleright M' & & \mathcal{I}' \triangleright N'
\end{tikzcd}$$

such that $\quad \mathcal{I}' \triangleright M' \approx_{bis} N'$

Typed Actions in DPI

$\boxed{\text{Actions must be \textbf{allowed} by environment}}$

Input: $\mathcal{I} \triangleright M \xrightarrow{k.a?V} \mathcal{I}, \mathcal{I}' \triangleright M'$ if

- $M \xrightarrow{k.a?V} M'$ - M can input on a at k
- $a : \mathsf{w}\langle \mathrm{T} \rangle_{@} k$ appears in $\mathcal{I}$
- $\mathcal{I}, \mathcal{I}' \vdash V : \mathrm{T}_{@} k$ - new names can be invented

Output: $\mathcal{I} \triangleright M \xrightarrow{(\tilde{c})k.a!V} \mathcal{I} \sqcap \langle V : \mathrm{T} \rangle_{@} k \ \triangleright \ M'$ if

- $M \xrightarrow{(\tilde{c})k.a!V} M'$ - M can output V on a at k
- $a : \mathsf{r}\langle \mathrm{T} \rangle_{@} k$ appears in $\mathcal{I}$

Making the connection

$\mathcal{I} \triangleright M$ is a configuration if

- $\exists \ \Delta <: \mathcal{I}$ such that - (Δ may allow more capabilities)
- $\Delta \vdash M$
- $\mathsf{dom}(\mathcal{I}) = \mathsf{dom}(\Delta)$

Thm: Configurations are preserved under typed actions

Thm: Over configurations $\approx_{bis}$ is preserved by contexts

Main Result

$\boxed{\text{Bisimulation Equivalence coincides with Contextual Equivalence}}$

For configurations: $\mathcal{I} \triangleright M \approx_{bis} N$ iff $\mathcal{I} \triangleright M \cong_{cxt} N$

Proof idea: For every μ and $\mathcal{I}$ there is a context $C[-]_\mu$ such that

$\mathcal{I} \triangleright M \stackrel{u}{\Longrightarrow} \mathcal{I}' \triangleright M'$

iff

- $\mathcal{I} \vdash C[-]_\mu$
- $C[M] \longrightarrow^* D[M', \mathcal{I}'] \mid \text{yeah!}\langle\rangle$

where $D[M', \mathcal{I}']$ *delivers* M' and $\mathcal{I}'$

Controlling Mobility - The taxman cometh

Problem: Knowledge of a name allows automatic access

Remote bank account server:

$$(\mathsf{newreg}\ \ put : \mathsf{rc}\langle \mathrm{T}_p\rangle,\ get : \mathsf{rc}\langle \mathrm{T}_g\rangle)$$

$$\mathit{Server} \Leftarrow s[\![\mathit{request}?(x : \mathbf{int}, y@z)\,(\mathsf{newloc}\, b : \mathrm{L}_b)\ \ \mathsf{with}\ \ldots$$

$$|\ \mathit{Taxman}\ \ ]\!]$$

$$\mathit{Taxman} \Leftarrow \mathit{deduct}?(x, \quad -\mathit{amount}$$

$$y, \quad -\mathit{bank\ account, type}\ \mathrm{L}_b$$

$$z) \quad -\mathit{ack}$$

$$\mathsf{goto}\, y.\ldots \mathit{collect\ tax\ with\ get, put}$$

Mobility Types - restricting access using *named locations*

Extended location types:

$$l : \mathsf{loc}[\mathbf{move_S}, \quad u_1 : \mathrm{A}_1, \ldots]$$

Only sites k in S have migration rights to l

Typing rule:

$$\frac{\begin{array}{l}\Gamma \vdash l[\![P]\!], \\ \Gamma \vdash l : \mathsf{loc}[\mathbf{move_{s \cup \{k\}}}]\end{array}}{\Gamma \vdash k[\![\mathsf{goto}\, l.P]\!]}$$

Note: Migration rights to l determined at *generation time*.

Avoiding the taxman

Server generates bank accounts with restricted migration rights.

$$(\mathsf{newreg}\ \ put : \mathsf{rc}\langle \mathrm{T}_p\rangle,\ get : \mathsf{rc}\langle \mathrm{T}_g\rangle)$$

$$\mathit{Server} \Leftarrow s[\![\mathit{request}?(x : \mathbf{int}, y@z, W) \qquad - W \ \mathit{allowed\ sites}$$

$$(\mathsf{newloc}\, b : \mathrm{L}_b^W)\ \ \mathsf{with}\ \ldots \mathit{code} \ldots]\!]$$

Declared type of new accounts L_b^W is

$$\mathsf{loc}[\mathbf{move}_{\mathbf{W}},\ \ put : \ldots get : \ldots]$$

Only locations specified at generation time, W, can access an account.

Note: L_b^W is a dynamic type.

Migration rights affect behavioural equivalences

Suppose Γ has no migration rights to site k:

$$\Gamma \triangleright k[\![b!\langle\rangle]\!] \quad \overset{?}{\cong}_{cxt} \quad k[\![\mathsf{stop}]\!]$$

Answer 1: Yes

Answer 2: No

Depends on definition of contextual equivalence $\cong_{cxt}$

Simple Mobility Rights

Restriction: only allow **universal** move capability $\mathbf{move}$ which grants access by **every** site.

$$l : \mathsf{loc}[\mathbf{move},\ \ u_1 : \mathrm{A}_1, \ldots] \quad \text{all sites have access to } l$$

$$k : \mathsf{loc}[u_1 : \mathrm{A}_1, \ldots] \qquad \text{no site has access to } k$$

Thm: yes $\boxed{\mathcal{I} \triangleright M \approx^{m}_{bis} N \text{ iff } \mathcal{I} \triangleright M \approx^{m}_{cxt} N}$

Thm: no $\boxed{\mathcal{I} \triangleright M \approx^{\mathcal{T}}_{bis} N \text{ is } \textbf{not} \text{ the same as } \mathcal{I} \triangleright M \approx^{\mathcal{T}}_{cxt} N}$

(For configurations)

Typed m-Bisimilarity $\approx^{m}_{bis}$ for mobility

Uses m-typed actions: $\boxed{\mathcal{I} \triangleright M \xrightarrow{\mu}_{m} \mathcal{I}' \triangleright M'}$

Input: $\mathcal{I} \triangleright M \xrightarrow{k.a?V}_{m} \mathcal{I}' \triangleright M'$ if

- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$
- M can input on a at k
- $\mathcal{I}$ allows it

Output: $\mathcal{I} \triangleright M \xrightarrow{(\tilde{c})a!V} \mathcal{I} \sqcap \langle V : \mathrm{T} \rangle_{@} k \;\triangleright\; M'$ if

- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$
- M can output V on a at k
- $\mathcal{I}$ allows it

Typed m-Contextual equivalence $\approx^m_{cxt}$

1. Replace clause:

- $\Gamma \triangleright M \cong_{cxt} N$
- $\Gamma \vdash O$

implies $\Gamma \triangleright O \mid M \cong_{cxt} O \mid N$

with

- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$
- $\Gamma \triangleright M \approx^m_{cxt} N$
- $\Gamma \vdash k[\![P]\!]$

implies $\Gamma \triangleright k[\![P]\!] \mid M \approx^m_{cxt} k[\![P]\!] \mid N$

2. Observations only allowed at sites t such that $\mathcal{I} \vdash t : \mathsf{loc}[\mathbf{move}]$

Typed $\mathcal{T}$-Contextual Equivalence $\approx^{\mathcal{T}}_{cxt}$

Let $\mathcal{T}$ be a set of sites at which the context has *apriori* processes running.

That is: $\approx^{\mathcal{T}}_{cxt}$ satisfies

- $\Gamma \triangleright M \approx^{\mathcal{T}}_{cxt} N$
- $\Gamma \vdash k[\![P]\!]$
- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$ or $k \in \mathcal{T}$

implies $\Gamma \triangleright k[\![P]\!] \mid M \approx^{\mathcal{T}}_{cxt} k[\![P]\!] \mid N$

Typed $\mathcal{T}$-Bisimilarity $\approx_{bis}^{\mathcal{T}}$

Uses $\mathcal{T}$-typed actions: $\boxed{\mathcal{I} \triangleright M \xrightarrow{\mu}_{\mathcal{T}} \mathcal{I}' \triangleright M'}$

Input: $\mathcal{I} \triangleright M \xrightarrow{k.a?V}_{\mathcal{T}} \mathcal{I}' \triangleright M'$ if

- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$ or $k \in \mathcal{T}$
- M can input on a at k
- $\mathcal{I}$ allows it

Output: $\mathcal{I} \triangleright M \xrightarrow{(\tilde{c})a!V} \mathcal{I} \sqcap \langle V : \mathrm{T} \rangle_{@} k \ \triangleright \ M'$ if

- $\ldots$

$$\mathcal{I} \triangleright M \approx^{\mathcal{T}}_{cxt} N \textbf{ does not imply } \mathcal{I} \triangleright M \approx^{\mathcal{T}}_{bis} N$$

Γ contains $h : \mathsf{loc}[\mathbf{move}],\ k : \mathsf{loc}[],\ a : \mathsf{rw}\langle \mathrm{T} \rangle_{@} h$

$\mathcal{T}$ contains k - context has already a process running at k

$$h[\![a!\langle b_{@}k\rangle]\!] \mid k[\![b!\langle\rangle]\!] \quad \approx^{\mathcal{T}}_{cxt} \quad h[\![a!\langle b_{@}k\rangle]\!] \mid k[\![\mathsf{stop}]\!]$$
$$\not\approx^{\mathcal{T}}_{bis}$$

Problem: With bisimulations information gained at h can be used at k, although context can not migrate from l to k.

Characterising $\mathcal{T}$-Contextual Equivalence $\approx^{\mathcal{T}}_{cxt}$

Thm: $\mathfrak{I} \triangleright M \approx^{\mathcal{T}}_{cxt} N$ iff $\mathfrak{I} \triangleright M \approx^{\mathcal{T}}_{bis} N$

$\mathfrak{I}$ contains:

- globally known capabilities
- capabilities known separately at each site in $\mathcal{T}$

Details: Watch this space

Further Work

- Write all this down
- Extend to selective capabilities move_S
- Allow dynamic update of migration rights
- Examine other ways of managing migration
- *all kinds of things*