Logical Properties of Name Restriction

Luca Cardelli
Andy Gordon

Microsoft Research

Semantics Lunch 2000-11-06

Properties of Secure Mobile Computation

- We would like to express properties of unique, private, hidden, and secret *names*:
  - "The applet is placed in a private sandbox."
  - "The key exchange happens in a secret location."
  - "A shared private key is established between two locations."
  - "A fresh nonce is generated and transmitted."
- Crucial to expressing this kind of properties is devising new logical quantifiers for *fresh* and *hidden* entities:
  - "There is a fresh (never used before) name such that …"
  - "There is a hidden (unnamable) location such that …"
  - N.B.: standard quantifiers are problematic. "There exists a sandbox containing the applet" is rather different from "There exists a fresh sandbox containing the applet" and from "There exists a hidden sandbox containing the applet".

Approach

- Use a specification logic grounded in an operational model of mobility. (So soundness is not an issue.)
- Express properties of dynamically changing structures of locations.
  - Previous work [POPL'00].
- Express properties of hidden names. We split it into two logical tasks:
  - Quantify over fresh names. We adopt [Gabbay-Pitts].
  - Reveal hidden names, so we can talk about them.
  - Combine the two, to quantify over hidden locations.

    "There is a hidden location …" represented as:

    "There is a fresh name that can be used to reveal (mention) the hidden name of a location …".

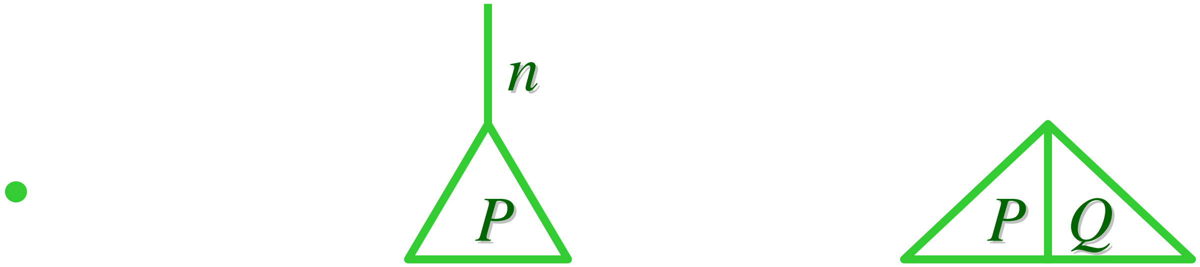
Spatial Logics

- We want to describe mobile behaviors. The *ambient calculus* provides an operational model, where spatial structures (agents, networks, etc.) are represented by nested locations.

- We also want to specify mobile behaviors. To this end, we devise an *ambient logic* that can talk about spatial structures.

Processes		*Formulas*	
$\mathbf{0}$	(void)	$\mathbf{0}$	(there is nothing here)
$n[P]$	(location)	$n[\mathcal{A}]$	(there is one thing here)
$P \mid Q$	(composition)	$\mathcal{A} \mid \mathcal{B}$	(there are two things here)

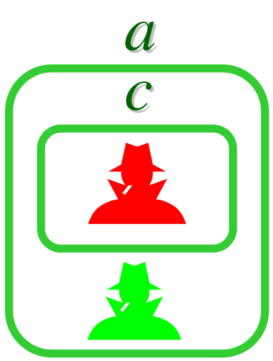
Trees

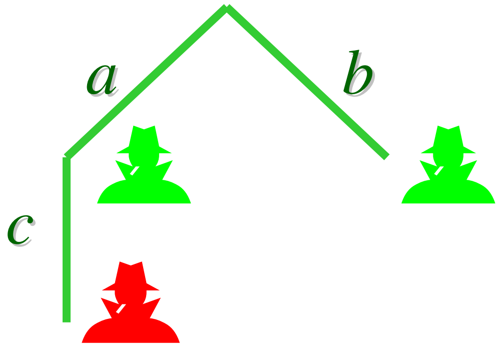


(void) (location) (composition)

Mobility

- *Mobility* is change of spatial structures over time.

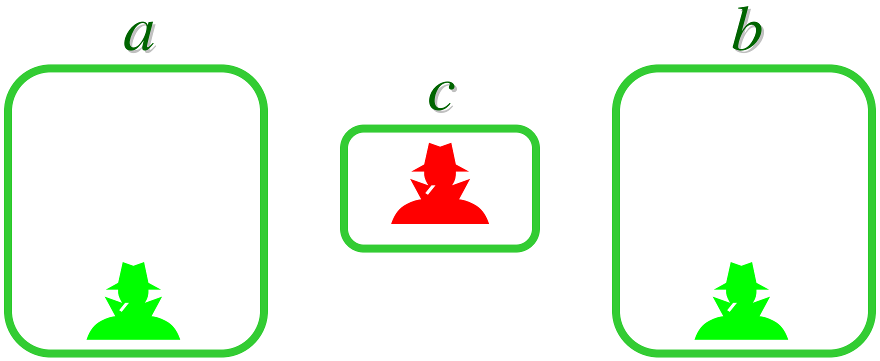


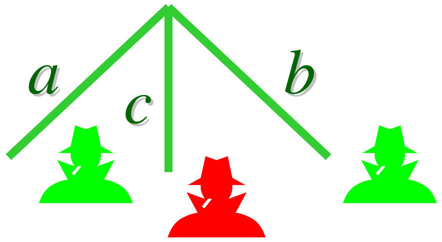


$a[Q \mid c[out\ a.\ in\ b.\ P]] \mid b[R]$

Mobility

- *Mobility* is change of spatial structures over time.

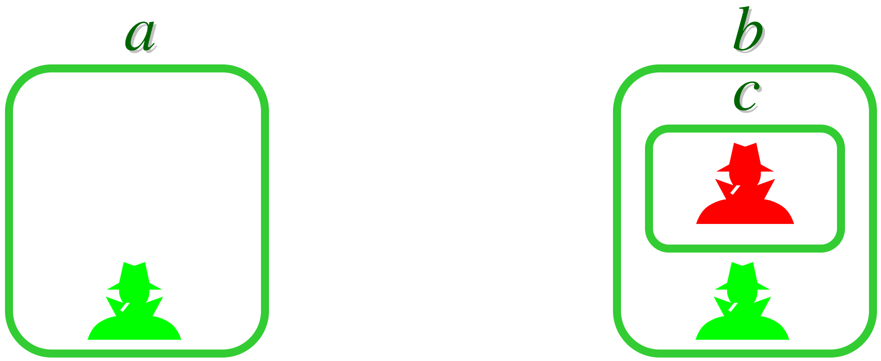


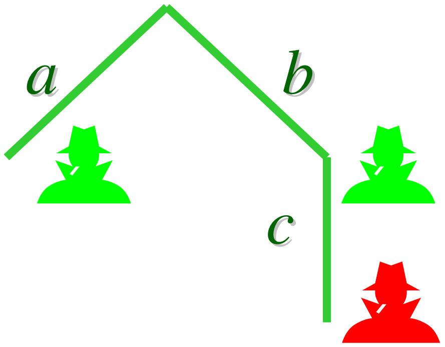


$$a[Q] \quad\quad \mid c[\mathit{in}\ b.\ P] \ \mid b[R]$$

Mobility

- *Mobility* is change of spatial structures over time.

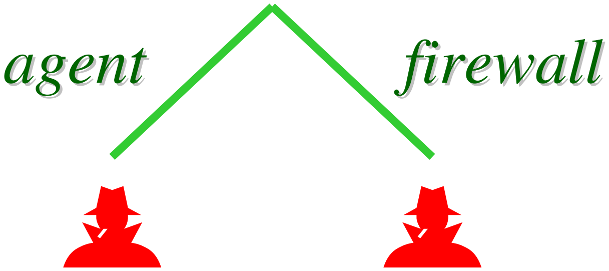




$a[Q] \mid b[R \mid c[P]]$

Properties of Mobile Computation

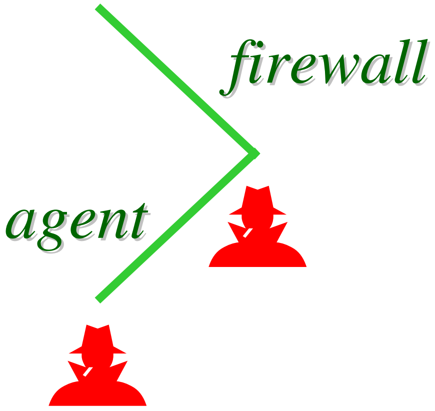
- These often have the form:
  - Right now, we have a spatial configuration, and later, we have another spatial configuration.
  - E.g.: Right now, the agent is outside the firewall, …



Now

Properties of Mobile Computation

- These often have the form:
  - Right now, we have a spatial configuration, and later, we have another spatial configuration.
  - E.g.: Right now, the agent is outside the firewall, and later (after running an authentication protocol), the agent is inside the firewall.



Later

Logical Formulas

$\mathcal{A} \in \Phi$ ::= Formulas (η is a name *n* or a variable *x*)

$\mathbf{T}$	true		
$\neg\mathcal{A}$	negation		
$\mathcal{A} \vee \mathcal{A}'$	disjunction		
$\mathbf{0}$	void		
$\eta[\mathcal{A}]$	location	$\mathcal{A}@\eta$	location adjunct
$\mathcal{A} \mid \mathcal{A}'$	composition	$\mathcal{A} \triangleright \mathcal{A}'$	composition adjunct
$\eta \circledR \mathcal{A}$	revelation	$\mathcal{A} \oslash \eta$	revelation adjunct
✧$\mathcal{A}$	somewhere modality		
$\Diamond\mathcal{A}$	sometime modality		
$\forall x.\mathcal{A}$	universal quantification over names		

Simple Examples

❶: $p[\mathbf{T}] \mid \mathbf{T}$

there is a location p here (and possibly something else)

❷: ✧❶

somewhere there is a location p

❸: ❷ $\Rightarrow$ $\Box$❷

if there is a p somewhere, then forever there is a p somewhere

❹: $p[q[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$

there is a p with a child q here

❺: ✧❹

somewhere there is a p with a child q

Intended Model: Ambient Calculus

$P \in \Pi ::=$	Processes	
$(\nu n)P$	restriction	Location Trees
$\mathbf{0}$	inactivity	
$P \mid P'$	parallel	
$M[P]$	ambient	
$!P$	replication	
$M.P$	exercise a capability	Actions
$(n).P$	input locally, bind to *n*	
$\langle M \rangle$	output locally (async)	

$M ::=$	Messages
n	name
$in\ M$	entry capability
$out\ M$	exit capability
$open\ M$	open capability
ε	empty path
$M.M'$	composite path

$n[] \triangleq n[\mathbf{0}]$

$M \triangleq M.\mathbf{0}$ (where appropriate)

Reduction Semantics

- A structural congruence relation $P \equiv Q$:
  - On spatial expressions, $P \equiv Q$ iff P and Q denote the same tree. So, the syntax modulo $\equiv$ is a notation for spatial trees.
  - On full ambient expressions, $P \equiv Q$ if in addition the respective threads are “trivially equivalent”.
  - Prominent in the definition of the logic.
- A reduction relation $P \longrightarrow^* Q$:
  - Defining the meaning of mobility and communication actions.
  - Closed up to structural congruence:

$$P \equiv P',\ P' \longrightarrow^* Q',\ Q' \equiv Q \quad \Rightarrow \quad P \longrightarrow^* Q$$

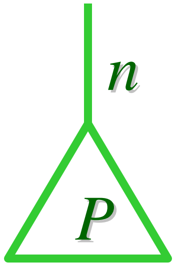
Meaning of Formulas: Satisfaction Relation

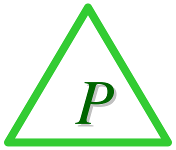
$$
\begin{array}{lll}
P \models \mathbf{T} & & \\
P \models \neg\mathcal{A} & \triangleq & \neg\, P \models \mathcal{A} \\
P \models \mathcal{A} \vee \mathcal{B} & \triangleq & P \models \mathcal{A} \vee P \models \mathcal{B} \\
P \models \mathbf{0} & \triangleq & P \equiv \mathbf{0} \\
P \models n[\mathcal{A}] & \triangleq & \exists P' \in \Pi.\ P \equiv n[P'] \wedge P' \models \mathcal{A} \\
P \models \mathcal{A}@n & \triangleq & n[P] \models \mathcal{A} \\
P \models \mathcal{A} \mid \mathcal{B} & \triangleq & \exists P', P'' \in \Pi.\ P \equiv P' \mid P'' \wedge P' \models \mathcal{A} \wedge P'' \models \mathcal{B} \\
P \models \mathcal{A} \triangleright \mathcal{B} & \triangleq & \forall P' \in \Pi.\ P' \models \mathcal{A} \Rightarrow P \mid P' \models \mathcal{B} \\
P \models n \circledR \mathcal{A} & \triangleq & \exists P' \in \Pi.\ P \equiv (\nu n)P' \wedge P' \models \mathcal{A} \\
P \models \mathcal{A} \oslash n & \triangleq & (\nu n)P \models \mathcal{A} \\
P \models \text{✧}\mathcal{A} & \triangleq & \exists P' \in \Pi.\ P \downarrow^{*} P' \wedge P' \models \mathcal{A} \\
P \models \Diamond\mathcal{A} & \triangleq & \exists P' \in \Pi.\ P \longrightarrow^{*} P' \wedge P' \models \mathcal{A} \\
P \models \forall x.\mathcal{A} & \triangleq & \forall m \in \Lambda.\ P \models \mathcal{A}\{x \leftarrow m\}
\end{array}
$$

$P \downarrow P'$ iff $\exists n, P''.\ P \equiv n[P'] \mid P''$; $\downarrow^{*}$ is the refl-trans closure of $\downarrow$

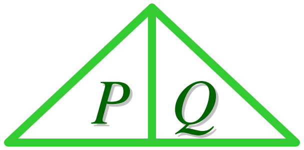
Satisfaction for Basic (rooted unordered edge-labeled finite-depth) Trees

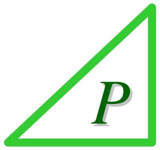
- $\models \mathbf{0}$

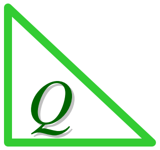


$\models n[\mathcal{A}]$ if 

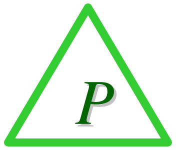
$\models \mathcal{A}$

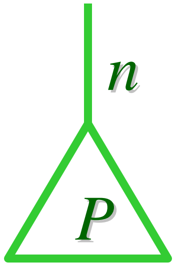


$\models \mathcal{A} \mid \mathcal{B}$ if 

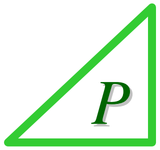
$\models \mathcal{A}$ and 

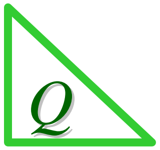
$\models \mathcal{B}$

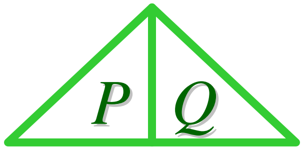


$\models \mathcal{A}@n$ if 

$\models \mathcal{A}$

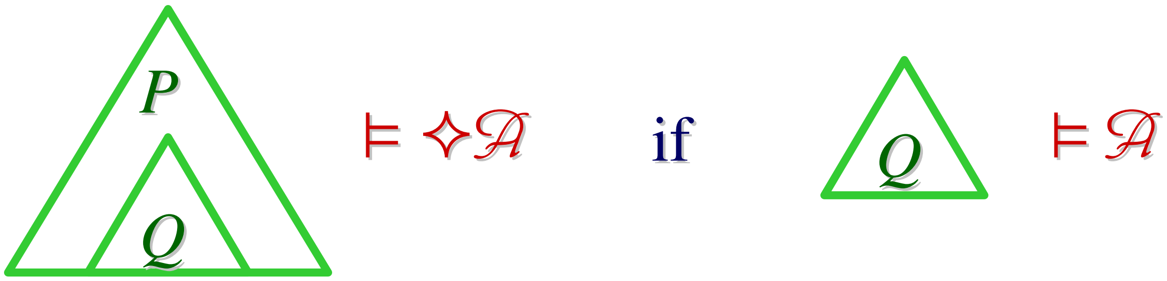


$\models \mathcal{A} \triangleright \mathcal{B}$ if for all 

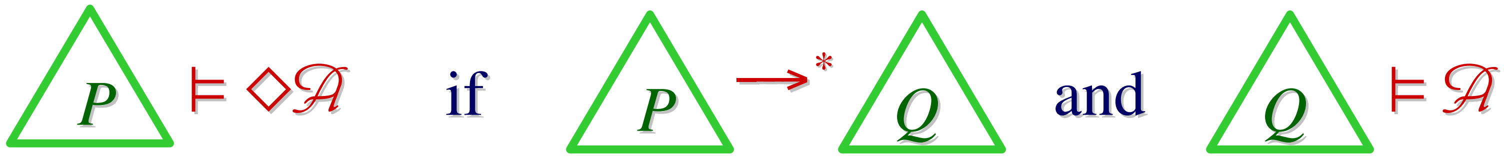
$\models \mathcal{A}$ we have 

$\models \mathcal{B}$

Satisfaction for Somewhere/Sometime



$\models ✧\mathcal{A}$ if $\models \mathcal{A}$

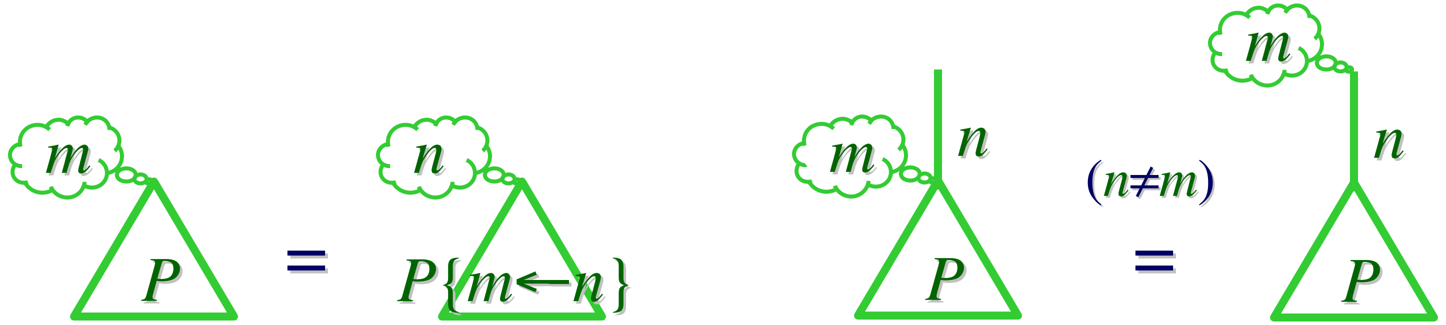


$\models \Diamond\mathcal{A}$ if $\longrightarrow^{*}$ and $\models \mathcal{A}$

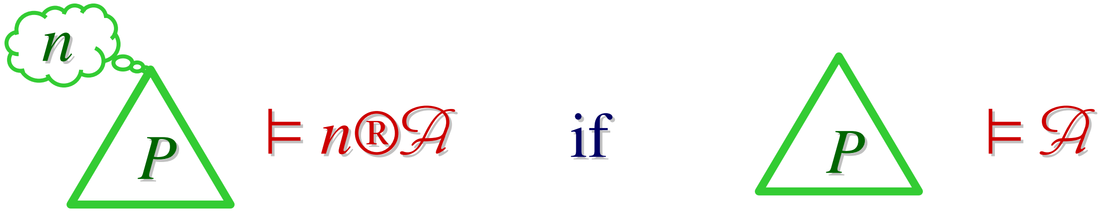
- N.B.: instead of $\Diamond\mathcal{A}$ and $✧\mathcal{A}$ we can use a “temporal next” operator $\circ\mathcal{A}$, along with the existing “spatial next” operator $n[\mathcal{A}]$, together with μ-calculus style recursive formulas.

Satisfaction for Revelation

- Trees with hidden labels:

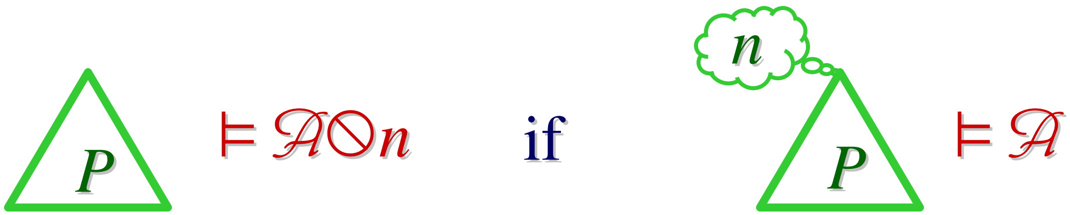


Etc.



⊨ n®$\mathcal{A}$ if ⊨ $\mathcal{A}$

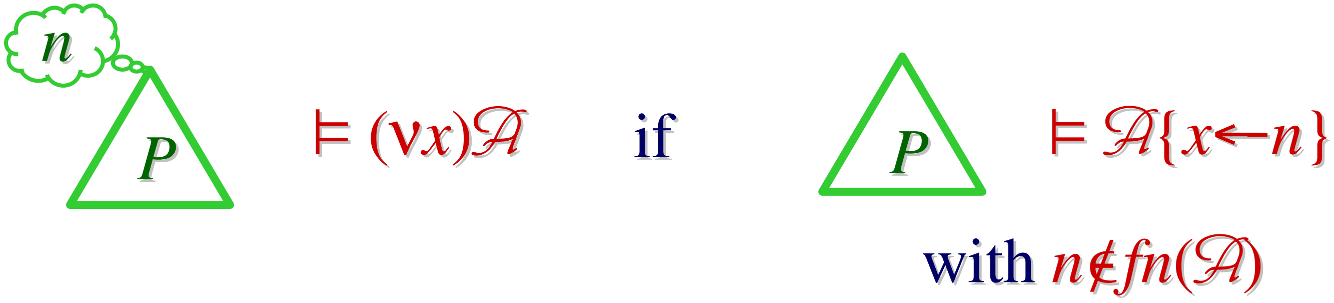
Not possible if
n is free!



⊨ $\mathcal{A}\oslash n$ if ⊨ $\mathcal{A}$

Hidden-Name Quantification

- Getting fancier:
  - $n \circledR \mathcal{A}$: reveal a hidden name if possible as n, and assert $\mathcal{A}\{n\}$.
  - $(\nu x)\mathcal{A}$: reveal a hidden name as any fresh name x and assert $\mathcal{A}\{x\}$.



$\models (\nu x)\mathcal{A}$ if $\models \mathcal{A}\{x \leftarrow n\}$

with $n \notin fn(\mathcal{A})$

- Design decision: how to define $(\nu x)\mathcal{A}$, keeping in mind that “freshness” may spill into the logic?
  - *The Obvious Thing*: extend the syntax with $(\nu x)\mathcal{A}$ and define it directly.
  - *Luis Caires:* Extend the syntax with $(\nu x)\mathcal{A}$ and add signatures to keep track of free names, to enforce the side condition $n \notin fn(\mathcal{A})$: $\Sigma \bullet P \models \Sigma \bullet \mathcal{A}$.
  - *Us:* Retain $n \circledR \mathcal{A}$ and mix it with a logical notions of freshness $\text{И}x.\mathcal{A}$ (one extra axiom schema, no new syntax). We eventually define: $(\nu x)\mathcal{A} \triangleq \text{И}x.x \circledR \mathcal{A}$.

Restriction (much as in the π-calculus)

- $(\nu n)P$
  - "The name n is known only inside P."
  - "Create a new name n and use it in P."
  - It *extrudes* (floats) because it represents knowledge, not behavior:

$(\nu n)P \equiv (\nu m)(P\{n \leftarrow m\})$

a private name is as good as another

$(\nu n)\mathbf{0} \equiv \mathbf{0}$

$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$

$(\nu n)(P \mid Q) \equiv (\nu n)P \mid Q$ if $n \notin fn(Q)$

a.k.a. $(\nu n)(P \mid (\nu n)Q') \equiv (\nu n)P \mid (\nu n)Q'$

scope extrusion

$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$

- Used initially to represent private channels.
- Later, to represent private names of any kind:

  Channels, Locations, Nonces, Cryptokeys, …

Revelation

$P \vDash n®\mathcal{A} \quad \triangleq \quad \exists P' \in \Pi.\ P \equiv (\nu n)P' \wedge P' \vDash \mathcal{A}$

- $n®\mathcal{A}$ is read, informally:
  - *Reveal* a private name as n and check that the revealed process satisfies $\mathcal{A}$.
  - Pull out (by extrusion) a (νn) binder, and check that the process stripped of the binder satisfies $\mathcal{A}$.
- Examples:
  - $n®n[\mathbf{0}]$: reveal a restricted name (say, p) as n and check the presence of an empty n location in the revealed process.

    $(\nu p)p[\mathbf{0}] \vDash n®n[\mathbf{0}]$

    because $(\nu p)p[\mathbf{0}] \equiv (\nu n)n[\mathbf{0}]$ and $n[\mathbf{0}] \vDash n[\mathbf{0}]$

Derived Formulas: Revelation

©*n*	≜ ¬*n*®T	*P* ⊨ - iff ¬∃*P*’∈Π. *P* ≡ (ν*n*)*P*’
		iff *n*∈*fn*(*P*)
closed	≜ ¬∃*x*.©*x*	*P* ⊨ - iff ¬∃*n*∈Λ. *n*∈*fn*(*P*)
separate	≜ ¬∃*x*.©*x* | ©*x*	*P* ⊨ - iff ¬∃*n*∈Λ, *P*’∈Π, *P*”∈Π.
		P ≡ *P*’ | *P*” ∧ *n*∈*fn*(*P*’) ∧ *n*∈*fn*(*P*”)

- Examples:
  - *n*[] ⊨ ©*n*
  - (ν*p*)*p*[] ⊨ *closed*
  - *n*[] | *m*[] ⊨ *separate*

Revelation Rules

- Some mirror properties of restriction:

  $\} \;\; x\circledR x\circledR\mathcal{A} \dashv\vdash x\circledR\mathcal{A}$

  $\} \;\; x\circledR y\circledR\mathcal{A} \dashv\vdash y\circledR x\circledR\mathcal{A}$

  $\} \;\; x\circledR(\mathcal{A} \mid x\circledR\mathcal{B}) \dashv\vdash x\circledR\mathcal{A} \mid x\circledR\mathcal{B}$ (scope extrusion)

- Some behave well with logical operators:

  $\} \;\; x\circledR(\mathcal{A} \vee \mathcal{B}) \vdash x\circledR\mathcal{A} \vee x\circledR\mathcal{A}$

  $\mathcal{A} \vdash \mathcal{B} \;\; \} \;\; x\circledR\mathcal{A} \vdash x\circledR\mathcal{B}$

- Some deal with the adjunction:

  $\eta\circledR\mathcal{A} \vdash \mathcal{B} \;\; \{\} \;\; \mathcal{A} \vdash \mathcal{B}\oslash\eta$

  $\} \;\; (\neg\mathcal{A})\oslash x \dashv\vdash \neg(\mathcal{A}\oslash x)$

  $\} \;\; (\mathcal{A} \mid \mathcal{B})\oslash x \vdash \mathcal{A}\oslash x \mid \mathcal{B}\oslash x$

  $\} \;\; x\circledR((\mathcal{A} \mid \mathcal{B})\oslash x) \dashv\vdash x\circledR(\mathcal{A}\oslash x) \mid x\circledR(\mathcal{B}\oslash x)$

Fresh-Name Quantifier

$P \vDash \text{И}x.\mathcal{A} \quad \triangleq \quad \exists m{\in}\Lambda.\ m{\notin}fn(P,\mathcal{A}) \wedge P \vDash \mathcal{A}\{x{\leftarrow}m\}$

- *C.f.*: $P \vDash \exists x.\mathcal{A}$ iff $\exists m{\in}\Lambda.\ P \vDash \mathcal{A}\{x{\leftarrow}m\}$
- Actually definable (metatheoretically, as an abbreviation):

  $\text{И}x.\mathcal{A} \ \triangleq\ \exists x.\ x\#(fnv(\mathcal{A})\text{-}\{x\}) \wedge x®\mathbf{T} \wedge \mathcal{A}$

  Provided we add the axiom schema:

  (GP) $\quad \} \ \exists x.\ x\#N \wedge x®\mathbf{T} \wedge \mathcal{A} \dashv\vdash \forall x.\ (x\#N \wedge x®\mathbf{T}) \Rightarrow \mathcal{A}$

  where $N \supseteq fnv(\mathcal{A})\text{-}\{x\}$ and $x{\notin}N$

- Fundamental "freshness" property (Gabbay-Pitts):

  $\text{И}x.\mathcal{A}$ iff $\exists m{\in}\Lambda.\ m{\notin}fn(P,\mathcal{A}) \wedge P \vDash \mathcal{A}\{x{\leftarrow}m\}$

  iff $\forall m{\in}\Lambda.\ m{\notin}fn(P,\mathcal{A}) \Rightarrow P \vDash \mathcal{A}\{x{\leftarrow}m\}$

  because *any fresh name as as good as any other*.

- Very nice logical properties:
  - $\forall x.\mathcal{A} \vdash \text{И}x.\mathcal{A} \vdash \exists x.\mathcal{A}$
  - $\neg\text{И}x.\mathcal{A} \dashv\vdash \text{И}x.\neg\mathcal{A}$
  - $\text{И}x.(\mathcal{A} \mid \mathcal{B}) \dashv\vdash (\text{И}x.\mathcal{A}) \mid (\text{И}x.\mathcal{B})$ (hint: (GP) $\exists$ for $\Rightarrow$, $\forall$ for $\Leftarrow$)
  - $\Diamond\text{И}x.\mathcal{A} \dashv\vdash \text{И}x.\Diamond\mathcal{A}$

Hidden-Name Quantifier

$(\nu x)\mathcal{A} \quad \triangleq \quad \text{И}x.x\circledR\mathcal{A}$

$P \vDash (\nu x)\mathcal{A}$ iff

$\exists m \in \Lambda, P' \in \Pi.\ m \notin fn(\mathcal{A}) \wedge P \equiv (\nu m)P' \wedge P' \vDash \mathcal{A}\{x \leftarrow m\}$

- Example: $(\nu x)x[\,] \;=\; \text{И}x.x\circledR x[\,]$
  - “for hidden x, we find a void location called x” = “for fresh x, we reveal a hidden name as x, then we find a void location x”
  - $(\nu n)n[\,] \vDash (\nu x)x[\,]$ because $(\nu n)n[\,] \vDash \text{И}x.x\circledR x[\,]$ because $(\nu n)n[\,] \vDash n\circledR n[\,]$ (where $n \notin fn((\nu n)n[\,])$).
- Counterexamples:
  - $(\nu m)m[\,] \nvDash (\nu x)n[\,]$ (N.B.: this holds for $(\nu x)\mathcal{A} \triangleq \exists x.x\circledR\mathcal{A}$!)
  - $(\nu n)n[\,] \mid (\nu n)n[\,] \nvDash (\nu x)(x[\,] \mid x[\,])$
  - $(\nu n)(n[\,] \mid n[\,]) \nvDash (\nu x)x[\,] \mid (\nu x)x[\,]$

A Good Property

- A property not shared by other candidate definitions, such as $\exists x.x\circledR\mathcal{A}$ and $\forall x.x\circledR\mathcal{A}$. This is even derivable within the logic:

$$(\nu x)(\mathcal{A}\{n\leftarrow x\}) \wedge n\circledR\mathbf{T} \dashv\vdash n\circledR\mathcal{A} \qquad \text{where } x \notin fv(\mathcal{A})$$

- It implies:

$$P \vDash \mathcal{A} \;\Rightarrow\; (\nu n)P \vDash (\nu x)(\mathcal{A}\{n\leftarrow x\})$$

$$P \vDash (\nu x)(\mathcal{A}\{n\leftarrow x\}) \wedge n\notin fn(P) \;\Rightarrow\; P \vDash n\circledR\mathcal{A}$$

$$P \vDash n\circledR\mathcal{A} \;\Rightarrow\; P \vDash (\nu x)(\mathcal{A}\{n\leftarrow x\})$$

A Surprising Property

$(\nu x)\mathcal{A} \nvdash \mathcal{A}$ for $x \notin fv(\mathcal{A})$

- Ex.: $(\nu x)(\neg\mathbf{0} \mid \neg\mathbf{0}) \nvdash \neg\mathbf{0} \mid \neg\mathbf{0}$

  If for a hidden x the inner system can be decomposed into two non-void parts, it does not mean that the whole system can be decomposed, because the two parts may be entangled by restriction:

  $(\nu n)(n[\,] \mid n[\,]) \vDash \text{И}x.x®(\neg\mathbf{0} \mid \neg\mathbf{0})$ but:

  $(\nu n)(n[\,] \mid n[\,]) \nvDash \neg\mathbf{0} \mid \neg\mathbf{0}$.
- This is ®’s fault, not И’s: with the same counterexample we can show $n®(\neg\mathbf{0} \mid \neg\mathbf{0}) \nvdash \neg\mathbf{0} \mid \neg\mathbf{0}$.
- However, $(\nu x)\mathbf{0} \vdash \mathbf{0}$.
- Moreover, $\mathcal{A} \vdash (\nu x)\mathcal{A}$ for $x \notin fv(\mathcal{A})$.

Forget n®$\mathcal{A}$ and И$x.\mathcal{A}$, why not just use $(\nu x)\mathcal{A}$?

- Consider:

  И$x.x$®$(\mathcal{A} \mid x$®$\mathcal{B})$

  $\dashv\vdash$ И$x.(x$®$\mathcal{A} \mid x$®$\mathcal{B})$

  $\dashv\vdash$ (И$x.x$®$\mathcal{A}) \mid ($И$x.x$®$\mathcal{B})$

- That is:

  $(\nu x)(\mathcal{A} \mid x$®$\mathcal{B}) \dashv\vdash (\nu x)\mathcal{A} \mid (\nu x)\mathcal{B}$

- Hence, the scope extrusion rule for (νx) still uses ®.
  - Can ® (or ©) be expressed via (νx)?
  - Is И useful if we have both ® and (νx)?

- In any case, we have explored interesting connections between these three operators.

Example: Key Sharing

- Consider a situation where “a hidden name x is shared by two locations n and m, and is not known outside those locations”.

$$(\nu x)\ (n[\copyright x] \mid m[\copyright x])$$

  - $P \models (\nu x)\ (n[\copyright x] \mid m[\copyright x])$

    $\Leftrightarrow \exists r \in \Lambda.\ r \notin fn(P) \cup \{n,m\} \wedge \exists R', R'' \in \Pi.\ P \equiv (\nu r)(n[R'] \mid m[R'']) \wedge r \in fn(R') \wedge r \in fn(R'')$

  - E.g.: take $P = (\nu p)\ (n[p[]] \mid m[p[]])$.

- A protocol establishing a shared key should satisfy:

$$\Diamond(\nu x)\ (n[\copyright x] \mid m[\copyright x])$$

Possible Applications

- Verifying security+mobility protocols.
- Modelchecking security+mobility assertions:
  - If P is $\mathbf{!}$-free and $\mathcal{A}$ is $\triangleright$-free, then $P \vDash \mathcal{A}$ is decidable.
  - This provides a way of mechanically checking (certain) assertions about (certain) mobile processes.
- Expressing mobility/security policies of host sites. (Conferring more flexibility than just sandboxing the agent.)
- Just-in-time verification of code containing mobility instructions (by either modelchecking or proof-carrying code).

Conclusions

- The novel aspects of our logic lie in its explicit treatment of space and of the evolution of space over time (mobility).
- We can now talk also about fresh and hidden locations.
- These ideas can be applied to any process calculus that embodies a distinction between spatial and temporal operators, and a restriction operator.
- Our logical rules arise from a particular model. This approach makes the logic very concrete (and sound), but raises questions of logical completeness.

<http://www.luca.demon.co.uk> Logical Properties of Name Restriction