Logics for Mobility

Luca Cardelli
Andy Gordon

Microsoft Research

Lausanne 2000-10-24

Simple Properties of Mobile Computation

- We have been looking for ways to express properties of mobile computations, E.g.:
  - "Here today, gone tomorrow."
  - "Eventually the agent crosses the firewall."
  - "Every agent carries a suitcase."
  - "Somewhere there is a virus."
  - "There is always at most one entity called *n* here."
- As with properties of ordinary concurrent computations, formalization options include:
  - Type systems (limited).
  - Equational reasoning (hard).
  - Reasoning on traces (ugly).
  - Reasoning via modal/temporal logics (a popular compromise).

Harder Properties

- Moreover, 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.)
- Find ways of expressing properties of dynamically changing structures of locations.
  - Previous work [POPL'00].
- Find ways of talking about hidden names. We split it into two logical tasks:
  - Find ways of quantifying over fresh names. We adopt a recent solution [Gabbay-Pitts].
  - Find ways of revealing hidden names, so we can talk about them.
  - Combine the two, to quantify over hidden locations.

    "There is a hidden location …" represented as:

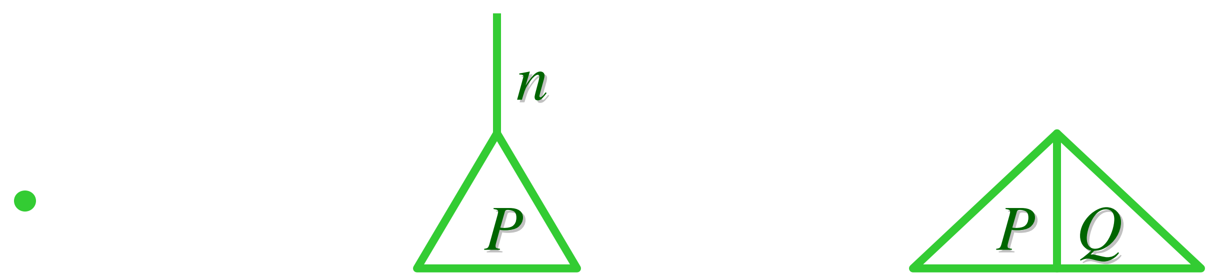
    "There is a fresh name that can be used to reveal (mention) the hidden name of a location …".

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)

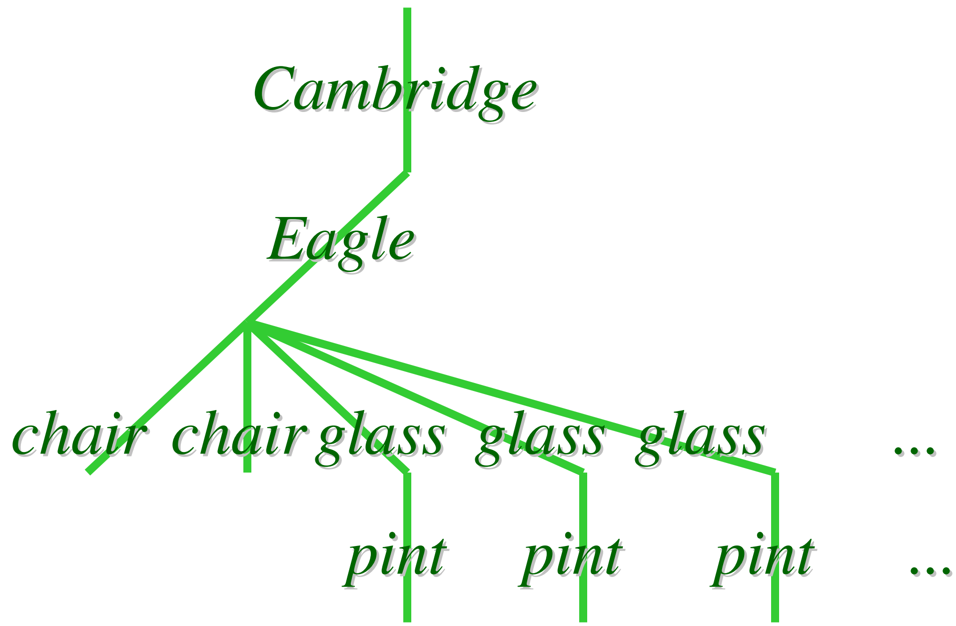
Trees



(void) (location) (composition)

Spatial Structures

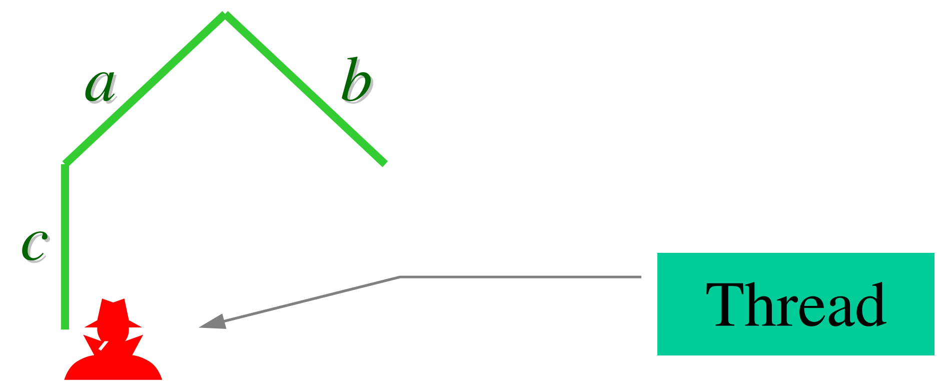
- Our basic model of space is going to be *finite-depth edge-labeled unordered trees* (*c.f.* semistructured data, XML). For short: *spatial trees*, represented by a syntax of *spatial expressions*. Unbounded resources are represented by infinite branching:



$$Cambridge[Eagle[chair[\mathbf{0}] \mid chair[\mathbf{0}] \mid \mathbf{!}glass[pint[\mathbf{0}]]] \mid ...]$$

Ambient Structures

- These spatial expressions/trees are a subset of ambient expressions/trees, which can represent both the spatial and the temporal aspects of mobile computation.

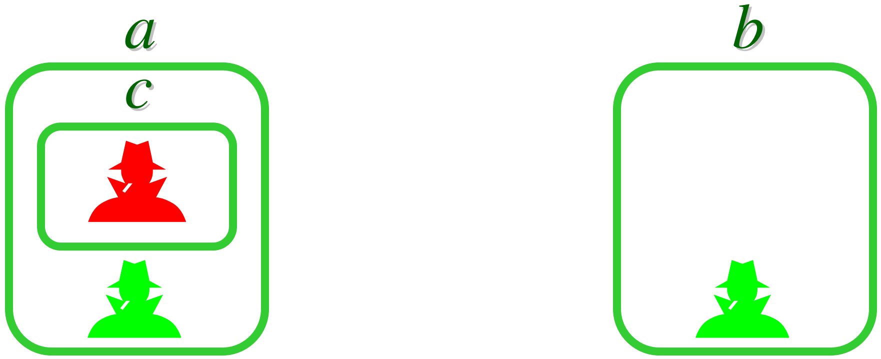


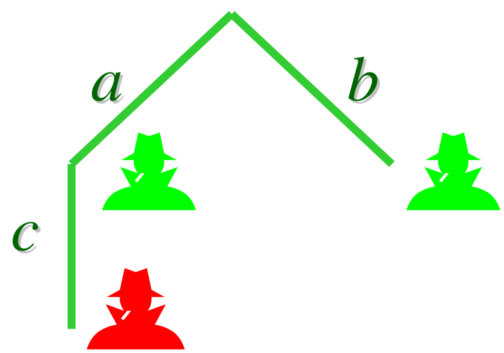
- An ambient tree is a spatial tree with, possibly, threads at each node that can locally change the shape of the tree.

$$a[c[\mathit{out}\ a.\ \mathit{in}\ b.\ P]] \mid b[\mathbf{0}]$$

Mobility

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

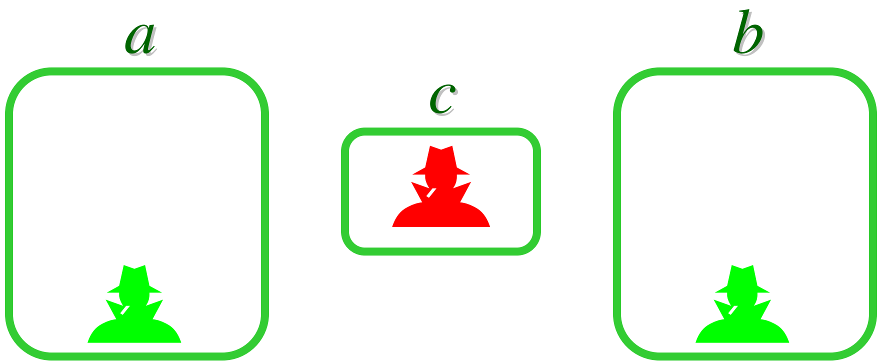


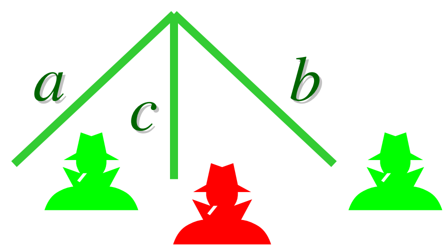


$a[Q \mid c[out\ a.\ in\ b.\ P]] \quad \mid b[R]$

Mobility

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

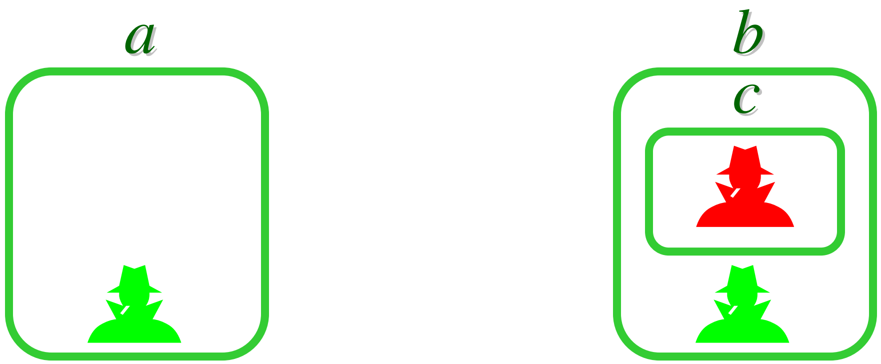


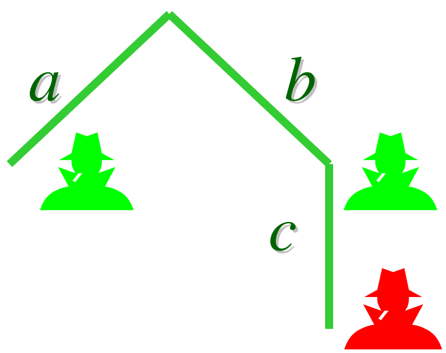


$$a[Q] \quad | \; c[in\ b.\ P] \;\; | \; b[R]$$

Mobility

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





$a[Q] \quad | \; b[R \mid c[P]]$

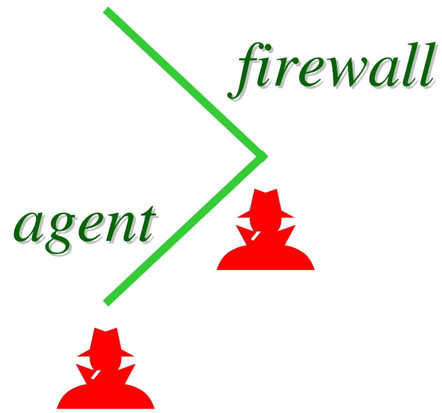
Properties of Mobile Computation

- 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

Ambient Calculus

$P \in \Pi ::=$	Processes	
$(\nu n)P$	restriction	
$\mathbf{0}$	inactivity	
$P \mid P'$	parallel	Trees
$M[P]$	ambient	
$!P$	replication	
$M.P$	exercise a capability	
$(n).P$	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:

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

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

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

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

$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad \text{if } n \notin fn(P)$ — scope extrusion

$(\nu n)(m[P]) \equiv m[(\nu n)P] \quad \text{if } n \neq m$

- Uses or restriction:
  - Initially to represent private channels.
  - Later, to represent private names of any kind:

    Channels, Locations, Nonces, Cryptokeys, …

Modal Logics

- In a modal logic, the truth of a formula is relative to a state (called a *world*).
  - Temporal logic: current time.
  - Program logic: current store contents.
  - Epistemic logic: current knowledge. Etc.
- In our case, the truth of a *space-time modal formula* is relative to the *here and now* of a process.
  - The formula $n[\mathbf{0}]$ is read:

    there is *here and now* an empty location called n
  - The operator $n[\mathcal{A}]$ is a single step in space (akin to the temporal next), which allows us talk about that place one step down into n.
  - Other modal operators talk about undetermined times (in the future) and undetermined places (in the location tree).

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
$\eta ® \mathcal{A}$	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: $\diamond$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: $\diamond$❹

somewhere there is a p with a child q

Satisfaction Relation

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

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

Basic Fact

- Satisfaction is invariant under structural congruence:

$$P \vDash \mathcal{A},\ P \equiv P' \quad \Rightarrow \quad P' \vDash \mathcal{A}$$

I.e.: $\{P \in \Pi \parallel P \vDash \mathcal{A}\}$ is closed under $\equiv$.

- Hence, formulas describe congruence-invariant properties.
  - In particular, formulas describe properties of spatial trees.
  - N.B.: Most process logics describe bisimulation-invariant properties.

Basic Tree Formulas

$$P \vDash \mathbf{0} \quad \triangleq \quad P \equiv \mathbf{0}$$

$$P \vDash n[\mathcal{A}] \quad \triangleq \quad \exists P' \in \Pi.\ P \equiv n[P'] \wedge P' \vDash \mathcal{A}$$

$$P \vDash \mathcal{A} \mid \mathcal{B} \quad \triangleq \quad \exists P', P'' \in \Pi.\ P \equiv P' \mid P'' \wedge P' \vDash \mathcal{A} \wedge P'' \vDash \mathcal{B}$$

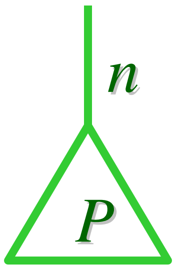
$$P \vDash \mathcal{A} @ n \quad \triangleq \quad n[P] \vDash \mathcal{A}$$

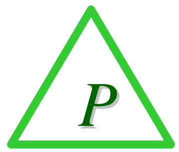
$$P \vDash \mathcal{A} \triangleright \mathcal{B} \quad \triangleq \quad \forall P' \in \Pi.\ P' \vDash \mathcal{A} \Rightarrow P \mid P' \vDash \mathcal{B}$$

- $\mathbf{0}$: there is no structure here now.
- $n[\mathcal{A}]$: there is a location n with contents satisfying $\mathcal{A}$.
- $\mathcal{A} \mid \mathcal{B}$: there are two structures satisfying $\mathcal{A}$ and $\mathcal{B}$.
- $\mathcal{A}@n$: when the current structure is placed in a location n, the resulting structure satisfies $\mathcal{A}$.
- $\mathcal{A} \triangleright \mathcal{B}$: when the current structure is composed with one satisfying $\mathcal{A}$, the resulting structures satisfies $\mathcal{B}$.

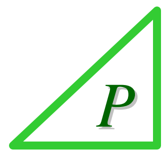
Satisfaction for Basic Trees

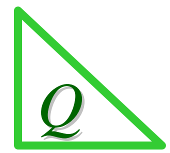
- $\vDash \mathbf{0}$



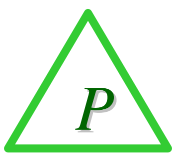
$\vDash n[\mathcal{A}]$ if


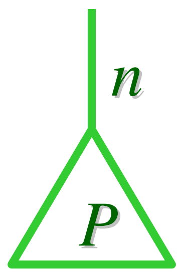
$\vDash \mathcal{A}$

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


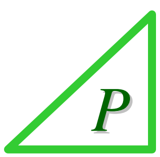
$\vDash \mathcal{A}$ and


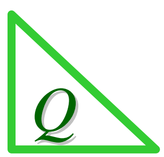
$\vDash \mathcal{B}$



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


$\vDash \mathcal{A}$

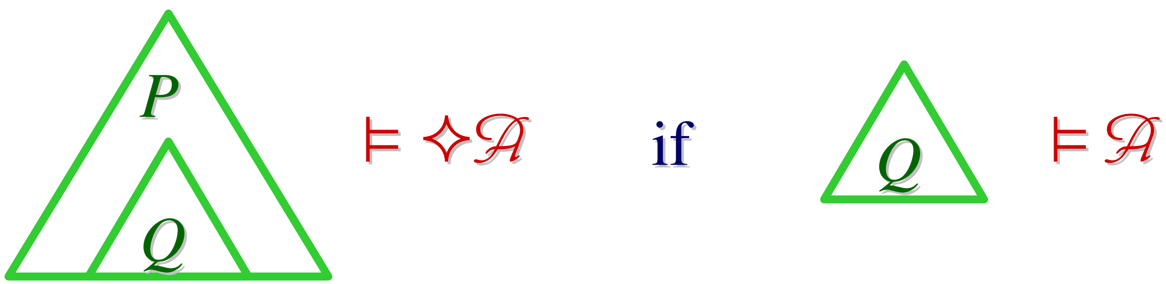


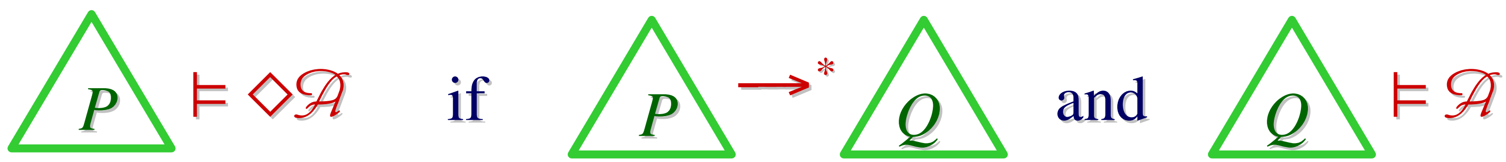
$\vDash \mathcal{A} \triangleright \mathcal{B}$ if for all


$\vDash \mathcal{A}$ we have


$\vDash \mathcal{B}$

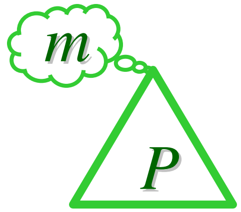
Satisfaction for Somewhere/Sometime



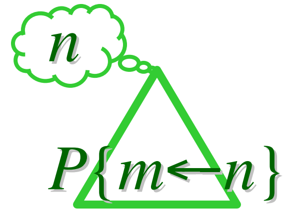


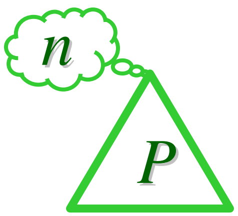
Satisfaction for Revelation

- Trees with hidden labels:

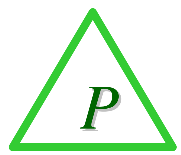


$=$

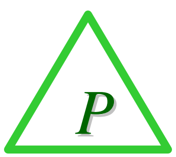




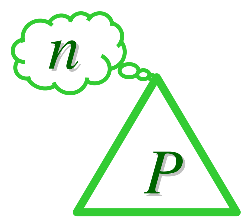
$\vDash n \circledR \mathcal{A}$ if



$\vDash \mathcal{A}$



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



$\vDash \mathcal{A}$

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

- $\mathbf{0} \vDash n$®$\mathbf{0}$ — because $\mathbf{0} \equiv (\nu n)\mathbf{0}$ and $\mathbf{0} \vDash \mathbf{0}$
- $m[\mathbf{0}] \vDash n$®$\mathbf{T}$ — because $m[\mathbf{0}] \equiv (\nu n)m[\mathbf{0}]$ and $m[\mathbf{0}] \vDash \mathbf{T}$
- $n[\mathbf{0}] \nvDash n$®$\mathbf{T}$ — because $n[\mathbf{0}] \not\equiv (\nu n)$...

- Therefore, the set of processes satisfying n®$\mathcal{A}$ is:
  - closed under α-variants
  - closed under $\equiv$-variants
  - not closed under changes in the set of free names
  - not closed under reduction (free names may disappear)
  - not closed under any equivalence that includes reduction
  - still ok for temporal reasoning: $\neg n$®$\mathcal{A} \wedge \Diamond n$®$\mathcal{A}$

Derived Formulas

$\mathbf{F}$	$\triangleq \neg\mathbf{T}$	
$\mathcal{A} \Rightarrow \mathcal{B}$	$\triangleq \neg\mathcal{A} \vee \mathcal{B}$	$P \vDash$ - iff $P \vDash \mathcal{A} \Rightarrow P \vDash \mathcal{B}$
$\mathcal{A} \wedge \mathcal{B}$	$\triangleq \neg(\neg\mathcal{A} \vee \neg\mathcal{B})$	$P \vDash$ - iff $P \vDash \mathcal{A} \wedge P \vDash \mathcal{B}$
$\exists x.\mathcal{A}$	$\triangleq \neg\forall x.\neg\mathcal{A}$	$P \vDash$ - iff $\exists m \in \Lambda.\ P \vDash \mathcal{A}\{x \leftarrow m\}$
$\Box\mathcal{A}$	$\triangleq \neg\Diamond\neg\mathcal{A}$	$P \vDash$ - iff $\forall P' \in \Pi.\ P \downarrow^* P' \Rightarrow P' \vDash \mathcal{A}$
¤$\mathcal{A}$	$\triangleq \neg$✧$\neg\mathcal{A}$	$P \vDash$ - iff $\forall P' \in \Pi.\ P \longrightarrow^* P' \Rightarrow P' \vDash \mathcal{A}$
$\mathcal{A}^{\mathbf{F}}$	$\triangleq \mathcal{A} \triangleright \mathbf{F}$	$P \vDash$ - iff $\forall P' \in \Pi.\ P' \vDash \mathcal{A} \Rightarrow P \mid P' \vDash \mathbf{F}$
		iff $\forall P' \in \Pi.\ \neg\ P' \vDash \mathcal{A}$
$\mathcal{A}^{\neg\mathbf{F}}$	$\mathcal{A}$ valid	$P \vDash$ - iff $\forall P' \in \Pi.\ P' \vDash \mathcal{A}$
$\mathcal{A}^{\mathbf{F}\neg}$	$\mathcal{A}$ satisfiable	$P \vDash$ - iff $\exists P' \in \Pi.\ P' \vDash \mathcal{A}$

Derived Formulas: Revelation

$\copyright n$	$\triangleq \neg n \circledR \mathbf{T}$	$P \vDash$ - iff $\neg\exists P' \in \Pi.\ P \equiv (\nu n)P'$
		iff $n \in fn(P)$
closed	$\triangleq \neg\exists x.\copyright x$	$P \vDash$ - iff $\neg\exists n \in \Lambda.\ n \in fn(P)$
separate	$\triangleq \neg\exists x.\copyright x \mid \copyright x$	$P \vDash$ - iff $\neg\exists n \in \Lambda,\ P' \in \Pi,\ P'' \in \Pi.$
		$P \equiv P' \mid P'' \wedge n \in fn(P') \wedge n \in fn(P'')$

- Examples:
  - $n[] \vDash \copyright n$
  - $(\nu p)p[] \vDash$ *closed*
  - $n[] \mid m[] \vDash$ *separate*

From Satisfaction to (Propositional) Logic

- Propositional validity

  $\boldsymbol{vld}\, \mathcal{A} \quad \triangleq \quad \forall P \in \Pi.\ P \vDash \mathcal{A}$ $\qquad\qquad$ $\mathcal{A}$ (closed) is valid

- Sequents

  $\mathcal{A} \vdash \mathcal{B} \quad \triangleq \quad \forall P \in \Pi.\ P \vDash \mathcal{A} \Rightarrow P \vDash \mathcal{B}$

- Rules

  $\mathcal{A}_1 \vdash \mathcal{B}_1; \ldots; \mathcal{A}_n \vdash \mathcal{B}_n \;\}\; \mathcal{A} \vdash \mathcal{B} \quad \triangleq$ $\qquad\qquad$ $(n \geq 0)$

  $\mathcal{A}_1 \vdash \mathcal{B}_1 \wedge \ldots \wedge \mathcal{A}_n \vdash \mathcal{B}_n \Rightarrow \mathcal{A} \vdash \mathcal{B}$

  (N.B.: all the rules shown later are validated accordingly.)

- Conventions:
  - $\dashv\vdash$ means $\vdash$ in both directions

    $\{\}$ means $\}$ in both directions

Omitted

- Logical axioms and rules.
  - Rules of propositional logic (standard).
  - Rules of location and composition

    $\mathcal{A} \mid \mathcal{C} \vdash \mathcal{B} \ \{\} \ \mathcal{A} \vdash \mathcal{C} \triangleright \mathcal{B}$ — $\mid$-$\triangleright$ adjunction
  - Rules of revelation

    $\eta ® \mathcal{A} \vdash \mathcal{B} \ \{\} \ \mathcal{A} \vdash \mathcal{B} \oslash \eta$ — ®-$\oslash$ adjunction

    $\} \ (\neg \mathcal{A}) \oslash x \dashv\vdash \neg(\mathcal{A} \oslash x)$ — $\oslash$ is self-dual
  - Rules of ✧ and ◇ modalities (standard S4, plus some)
  - Rules of quantification (standard, but for name quantifiers)
- A large collection of logical consequences.

Ex: Immovable Object vs. Irresistible Force

$$Im \triangleq \mathbf{T} \triangleright \Box(obj[] \mid \mathbf{T})$$

$$Ir \triangleq \mathbf{T} \triangleright \Box\Diamond\neg(obj[] \mid \mathbf{T})$$

$Im \mid Ir \vdash (\mathbf{T} \triangleright \Box(obj[] \mid \mathbf{T})) \mid \mathbf{T}$	$\mathcal{A} \vdash \mathbf{T}$
$\vdash \Box(obj[] \mid \mathbf{T})$	$(\mathcal{A} \triangleright \mathcal{B}) \mid \mathcal{A} \vdash \mathcal{B}$
$\vdash \Diamond\Box(obj[] \mid \mathbf{T})$	$\mathcal{A} \vdash \Diamond\mathcal{A}$
$Im \mid Ir \vdash \mathbf{T} \mid (\mathbf{T} \triangleright \Box\Diamond\neg(obj[] \mid \mathbf{T}))$	$\mathcal{A} \vdash \mathbf{T}$
$\vdash \Box\Diamond\neg(obj[] \mid \mathbf{T})$	$\Diamond\neg\mathcal{A} \vdash \neg\Box\mathcal{A}$
$\vdash \neg\Diamond\Box(obj[] \mid \mathbf{T})$	$\Box\neg\mathcal{A} \vdash \neg\Diamond\mathcal{A}$

Hence: $Im \mid Ir \vdash \mathbf{F}$ $\quad$ $\mathcal{A} \wedge \neg\mathcal{A} \vdash \mathbf{F}$

Example: Thief!

- A *shopper* is likely to pull out a wallet. A *thief* is likely to grab it.

$$Shopper \triangleq$$
$$Person[Wallet[\$] \mid \mathbf{T}] \wedge$$
$$\Diamond(Person[NoWallet] \mid Wallet[\$])$$

$$NoWallet \triangleq \neg(Wallet[\$] \mid \mathbf{T})$$

$$Thief \triangleq Wallet[\$] \triangleright \Diamond NoWallet$$

- By simple logical deductions involving laws of $\triangleright$ and $\Diamond$:

$$Shopper \mid Thief \Rightarrow$$
$$(Person[Wallet[\$] \mid \mathbf{T}] \mid Thief) \wedge$$
$$\Diamond(Person[NoWallet] \mid NoWallet)$$

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):

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

- 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 properties:
  - $\forall x.\mathcal{A} \Rightarrow \text{И}x.\mathcal{A} \Rightarrow \exists x.\mathcal{A}$

    $\neg\text{И}x.\mathcal{A} \Leftrightarrow \text{И}x.\neg\mathcal{A}$
  - $\text{И}x.(\mathcal{A} \mid \mathcal{B}) \Leftrightarrow (\text{И}x.\mathcal{A}) \mid (\text{И}x.\mathcal{B})$
  - $\Diamond\text{И}x.\mathcal{A} \Leftrightarrow \text{И}x.\Diamond\mathcal{A}$

Hidden-Name Quantifier

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

- Example: $(\nu x)x[\mathbf{T}] \;=\; \text{И} x.x®x[\mathbf{T}]$
  - "for hidden x, we find a location called x" = "for fresh x, we reveal a hidden name as x, then we find a location called x"
  - $(\nu n)n[] \vDash (\nu x)x[\mathbf{T}]$ because $(\nu n)n[] \vDash \text{И} x.x®x[\mathbf{T}]$ because $(\nu n)n[] \vDash n®n[\mathbf{T}]$ (where $n \notin fn((\nu n)n[])$).
- Other examples
  - $(\nu m)m[] \vDash (\nu x)n[]$
  - $(\nu n)n[] \mid (\nu n)n[] \nvDash (\nu x)(x[] \mid x[])$
  - $(\nu n)(n[] \mid n[]) \nvDash (\nu x)x[] \mid (\nu x)x[]$

A Good Property

- A property not shared by other candidate definitions (it is even derivable within the logic):

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

It implies:

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

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

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

Example: Key Sharing

- Consider a situation where "a hidden name x is shared by two locations n and m, and is <u>not known</u> outside those locations".

$$(\nu x)\ (n[©x] \mid m[©x])$$

  - $P \vDash (\nu x)\ (n[©x] \mid m[©x])$

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

- A protocol establishing a shared key should satisfy:

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

Applications

- Verifying security+mobility protocols.
- Modelchecking security+mobility assertions:
  - If P is $!$-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). The logic has a linear flavor in the sense that space cannot be instantly created or deleted, although it can be transformed over time.
- These ideas can be applied to any process calculus that embodies a distinction between spatial and temporal operators.
- Our logical rules arise from a particular model. This approach makes the logic very concrete (and sound), but raises questions of logical completeness, which are being investigated.