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

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(void)</td>
</tr>
<tr>
<td>n[P]</td>
<td>(location)</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
</tr>
</tbody>
</table>

### Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(there is nothing here)</td>
</tr>
<tr>
<td>n[A]</td>
<td>(there is one thing here)</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

### 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  chair  glass  glass  glass ...
        pint  pint  pint ...
Cambridge[Eagle[chair[0] | chair[0] | !glass[pint[0]]] | ...]
```
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.

\[ a[c[out \ a. \ in \ b. \ P]] \mid b[0] \]
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.

\[ a[Q] \quad | \quad c[\text{in } b. \ P] \quad | \quad b[R] \]
Mobility

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

\[
a[Q] \quad \mid b[R \mid c[P]]
\]
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, …
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.
### Ambient Calculus

<table>
<thead>
<tr>
<th>$P \in \Pi ::= $ Processes</th>
<th>$M ::= $ Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\forall n)P$ restriction</td>
<td>$n$ name</td>
</tr>
<tr>
<td>$0$ inactivity</td>
<td>$in M$ entry capability</td>
</tr>
<tr>
<td>$P</td>
<td>P'$ parallel</td>
</tr>
<tr>
<td>$M[P]$ ambient</td>
<td>$open M$ open capability</td>
</tr>
<tr>
<td>$!P$ replication</td>
<td>$\epsilon$ empty path</td>
</tr>
<tr>
<td>$M.P$ exercise a capability</td>
<td>$M.M'$ composite path</td>
</tr>
<tr>
<td>$(n).P$ input locally, bind to $n$</td>
<td></td>
</tr>
<tr>
<td>$\langle M \rangle$ output locally (async)</td>
<td></td>
</tr>
</tbody>
</table>

- $n[] \triangleq n[0]$
- $M \triangleq M.0$ (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 $P \rightarrow^* Q$:
  • Defining the meaning of mobility and communication actions.
  • Closed up to structural congruence:

\[
P \equiv P', \ P' \rightarrow^* Q', \ Q' \equiv Q \quad \Rightarrow \quad P \rightarrow^* Q
\]
Restriction (much as in the $\pi$-calculus)

- $(\forall n)P$
  - “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:

  $$(\forall n)P \equiv (\forall m)(P\{n\leftarrow m\})$$
  $$(\forall n)0 \equiv 0$$
  $$(\forall n)(\forall m)P \equiv (\forall m)(\forall n)P$$
  $$(\forall n)(P \mid Q) \equiv P \mid (\forall n)Q \quad \text{if } n \notin fn(P)$$
  $$(\forall n)(m[P]) \equiv m[(\forall n)P] \quad \text{if } n \neq m$$

- 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 $n[0]$ is read:
    
    there is *here and now* an empty location called $n$
  
  - The operator $n[A]$ 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

$A \in \Phi ::= \begin{align*}
&\text{Formulas} & (\eta \text{ is a name } n \text{ or a variable } x) \\
&T & \text{true} \\
&\neg A & \text{negation} \\
&A \lor A' & \text{disjunction} \\
&0 & \text{void} \\
&\eta[A] & \text{location} \\
&A \mid A' & \text{composition} \\
&\eta\circ A & \text{revelation} \\
&\text{somewhere modality} \\
&\Diamond A & \text{sometime modality} \\
&\forall x. A & \text{universal quantification over names} \\
&A@\eta & \text{location adjunct} \\
&A\circ\eta & \text{composition adjunct} \\
&A\circ\eta & \text{revelation adjunct} \\
\end{align*}$
Simple Examples

1: \( p[T] \mid T \)

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

2: \( \diamond 1 \)

somewhere there is a location \( p \)

3: \( 2 \Rightarrow \square 2 \)

if there is a \( p \) somewhere, then forever there is a \( p \) somewhere

4: \( p[q[T] \mid T] \mid T \)

there is a \( p \) with a child \( q \) here

5: \( \diamond 4 \)

somewhere there is a \( p \) with a child \( q \)
### Satisfaction Relation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \models T$</td>
<td>$\triangleleft \neg P \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \neg \mathcal{A}$</td>
<td>$\triangleleft \neg P \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \lor B$</td>
<td>$\triangleleft P \models \mathcal{A} \lor B$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \land B$</td>
<td>$\triangleleft P \models \mathcal{A} \land B$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \rightarrow B$</td>
<td>$\triangleleft \forall P \in \Pi. P \models \mathcal{A} \Rightarrow P \models B$</td>
</tr>
</tbody>
</table>
| $P \models \mathcal{A} 
abla B$ | $\triangleleft \exists P, P'' \in \Pi. P \equiv P' \land P'' \models \mathcal{A} \land P'' \models B$ |
| $P \models \mathcal{A} \cap B$ | $\triangleleft \exists P \in \Pi. P \models \mathcal{A} \land P \models B$ |
| $P \models \mathcal{A} \cup B$ | $\triangleleft (\forall n)P' \land P' \models \mathcal{A}$ |
| $P \models \mathcal{A} \setminus n$ | $\triangleleft \exists P \in \Pi. P \models \mathcal{A}$ |
| $P \models \mathcal{A} \Re n$ | $\triangleleft (\forall n)P \models \mathcal{A}$ |
| $P \models \mathcal{A} \oplus B$ | $\triangleleft \exists P \in \Pi. P \downarrow *P' \land P' \models \mathcal{A}$ |
| $P \models \mathcal{A} \bigotimes B$ | $\triangleleft \exists P \in \Pi. P \rightarrow *P' \land P' \models \mathcal{A}$ |
| $P \models \forall x. \mathcal{A}$ | $\forall m \in \Lambda. P \models \mathcal{A}\{x \leftarrow m\}$ |

$P \downarrow P'$ 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:

\[ P \models \mathcal{A}, \quad P \equiv P' \quad \Rightarrow \quad P' \models \mathcal{A} \]

I.e.: \( \{P \in \Pi \mid P \models \mathcal{A}\} \) 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

<table>
<thead>
<tr>
<th>Formula</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \vdash 0 \triangleq P \equiv 0$</td>
<td>0: there is no structure here now.</td>
</tr>
<tr>
<td>$P \vdash n[\mathcal{A}] \triangleq \exists P' \in \Pi. P \equiv n[P'] \land P' \vdash \mathcal{A}$</td>
<td>$n[\mathcal{A}]$: there is a location $n$ with contents satisfying $\mathcal{A}$.</td>
</tr>
<tr>
<td>$P \vdash \mathcal{A} \mid B \triangleq \exists P', P'' \in \Pi. P \equiv P' \mid P'' \land P' \vdash \mathcal{A} \land P'' \vdash B$</td>
<td>$\mathcal{A} \mid B$: there are two structures satisfying $\mathcal{A}$ and $B$.</td>
</tr>
<tr>
<td>$P \vdash \mathcal{A} \bowtie n \triangleq n[P] \vdash \mathcal{A}$</td>
<td>$\mathcal{A} \bowtie n$: when the current structure is placed in a location $n$, the resulting structure satisfies $\mathcal{A}$.</td>
</tr>
<tr>
<td>$P \vdash \mathcal{A} \triangleright B \triangleq \forall P' \in \Pi. P' \vdash \mathcal{A} \Rightarrow P \mid P' \vdash B$</td>
<td>$\mathcal{A} \triangleright B$: when the current structure is composed with one satisfying $\mathcal{A}$, the resulting structures satisfies $B$.</td>
</tr>
</tbody>
</table>
Satisfaction for Basic Trees

- $\models 0$

\[ n \models n[\mathcal{A}] \quad \text{if} \quad P \models \mathcal{A} \]

\[ P \quad Q \models \mathcal{A} \land \mathcal{B} \quad \text{if} \quad P \models \mathcal{A} \quad \text{and} \quad Q \models \mathcal{B} \]

\[ P \models \mathcal{A}@n \quad \text{if} \quad n \models \mathcal{A} \]

\[ P \models \mathcal{A} \supset \mathcal{B} \quad \text{if for all} \quad Q \models \mathcal{A} \quad \text{we have} \quad P \quad Q \models \mathcal{B} \]
Satisfaction for Somewhere/Sometime

\begin{align*}
\models \Diamond \mathcal{A} \quad \text{if} \quad \models \mathcal{A}\\
\models \Diamond \mathcal{A} \quad \text{if} \quad P \rightarrow^* Q \quad \text{and} \quad \models \mathcal{A}
\end{align*}
Satisfaction for Revelation

- Trees with hidden labels:

\[
P \\ P\{m\leftarrow n\}
\]

\[
P \models n \circ A \quad \text{if} \quad P \models A
\]

\[
P \models A \sqcap n \quad \text{if} \quad P \models A
\]
Revelation

\[ P \equiv n \otimes \bar{A} \iff \exists P' \in \Pi. P \equiv (\forall n) P' \land P' \equiv A \]

- \( n \otimes \bar{A} \) is read, informally:
  - \textit{Reveal} a private name as \( n \) and check that the revealed process satisfies \( \bar{A} \).
  - Pull out (by extrusion) a \((\forall n)\) binder, and check that the process stripped of the binder satisfies \( \bar{A} \).

- Examples:
  - \( n \otimes n[0] \): reveal a restricted name (say, \( p \)) as \( n \) and check the presence of an empty \( n \) location in the revealed process.
    \[
    (\forall p)p[0] \equiv n \otimes n[0]
    \]
    because \((\forall p)p[0] \equiv (\forall n)n[0]\) and \( n[0] \equiv n[0]\)
• $0 \vdash n \circ 0$ because $0 \equiv (\forall n)0$ and $0 \vdash 0$
• $m[0] \vdash n \circ \top$ because $m[0] \equiv (\forall n)m[0]$ and $m[0] \vdash \top$
• $n[0] \not\equiv n \circ \top$ because $n[0] \not\equiv (\forall n)…$

• Therefore, the set of processes satisfying $n \circ \mathcal{A}$ is:
  • closed under $\alpha$-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: $\neg n \circ \mathcal{A} \land \Diamond n \circ \mathcal{A}$
# Derived Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Definition</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( \equiv \neg T )</td>
<td>( P \vdash\text{iff } P \vdash A \Rightarrow P \vdash B )</td>
</tr>
<tr>
<td>( A \Rightarrow B )</td>
<td>( \equiv \neg A \vee B )</td>
<td>( P \vdash\text{iff } P \vdash A \wedge P \vdash B )</td>
</tr>
<tr>
<td>( A \wedge B )</td>
<td>( \equiv \neg (\neg A \vee \neg B) )</td>
<td>( P \vdash\text{iff } \exists m \in \Lambda. P \vdash A[x \leftarrow m] )</td>
</tr>
<tr>
<td>( \exists x.A )</td>
<td>( \equiv \neg \forall x.\neg A )</td>
<td>( P \vdash\text{iff } P \vdash \forall P' \in \Pi. P \downarrow *P' \Rightarrow P' \vdash A )</td>
</tr>
<tr>
<td>( \Box A )</td>
<td>( \equiv \neg \Diamond \neg A )</td>
<td>( P \vdash\text{iff } P \vdash \forall P' \in \Pi. P \rightarrow *P' \Rightarrow P' \vdash A )</td>
</tr>
<tr>
<td>( \forall A )</td>
<td>( \equiv \neg \forall \Diamond \neg A )</td>
<td>( P \vdash\text{iff } P \vdash \forall P' \in \Pi. P \downarrow *P' \Rightarrow \forall P' \vdash A )</td>
</tr>
<tr>
<td>( A \forall )</td>
<td>( \equiv A \Rightarrow F )</td>
<td>( P \vdash\text{iff } P \vdash \forall P' \in \Pi. P \vdash A \Rightarrow P \downarrow P' \vdash F )</td>
</tr>
<tr>
<td>( A \rightarrow \neg F )</td>
<td>( \equiv A \text{ valid} )</td>
<td>( P \vdash\text{iff } P \vdash \forall P' \in \Pi. P \vdash A )</td>
</tr>
<tr>
<td>( A \neg )</td>
<td>( \equiv A \text{ satisfiable} )</td>
<td>( P \vdash\text{iff } \exists P' \in \Pi. P \vdash A )</td>
</tr>
</tbody>
</table>
Derived Formulas: Revelation

\[ \diamond n \triangleq \neg n \otimes T \quad P \models \iff \neg \exists P' \in \Pi. P \equiv (\forall n)P' \]
\[ \text{iff } n \in \text{fn}(P) \]

\[ \text{closed} \triangleq \neg \exists x. \diamond x \quad P \models \iff \neg \exists n \in \Lambda. n \in \text{fn}(P) \]

\[ \text{separate} \triangleq \neg \exists x. \diamond x \mid \diamond x \quad P \models \iff \neg \exists n \in \Lambda, P' \in \Pi, P'' \in \Pi. \]
\[ P \equiv P' \mid P'' \land n \in \text{fn}(P') \land n \in \text{fn}(P'') \]

• **Examples:**
  
  • \( n[] \models \diamond n \)
  
  • \( (\forall p)p[] \models \text{closed} \)
  
  • \( n[] \mid m[] \models \text{separate} \)
From Satisfaction to (Propositional) Logic

- **Propositional validity**
  \[ \text{vld } \mathcal{A} \triangleq \forall P \in \Pi. P \vdash \mathcal{A} \]
  \(\mathcal{A}\) (closed) is valid

- **Sequents**
  \[ \mathcal{A} \vdash B \triangleq \forall P \in \Pi. P \vdash \mathcal{A} \Rightarrow P \vdash B \]

- **Rules**
  \[ \mathcal{A}_1 \vdash B_1; \ldots; \mathcal{A}_n \vdash B_n \} \mathcal{A} \vdash B \triangleq (n \geq 0) \]
  \[ \mathcal{A}_1 \vdash B_1 \land \ldots \land \mathcal{A}_n \vdash B_n \Rightarrow \mathcal{A} \vdash B \]
  (N.B.: all the rules shown later are validated accordingly.)

- **Conventions:**
  - \(\vdash\) means \(\vdash\) in both directions
  - \}\ means \}\ in both directions
• Logical axioms and rules.
  • Rules of propositional logic (standard).
  • Rules of location and composition
    \[ A \mid C \vdash B \quad \Rightarrow \quad A \vdash C \triangleright B \]  
    \( \triangleright \) adjunction
  • Rules of revelation
    \[ \eta \circ A \vdash B \quad \Rightarrow \quad A \vdash B \otimes \eta \]  
    \( \circ \)-\( \otimes \) adjunction
    \( \otimes \) is self-dual
  • Rules of \( \diamond \) and \( \Diamond \) 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

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

\[ Im \mid Ir \vdash (T \triangleright \Box (\text{obj[]} \mid T)) \mid T \]

\[ \vdash \Box (\text{obj[]} \mid T) \]

\[ \vdash \Diamond \Box (\text{obj[]} \mid T) \]

\[ Im \mid Ir \vdash T \mid (T \triangleright \Box \Diamond \neg (\text{obj[]} \mid T)) \]

\[ \vdash \Box \Diamond \neg (\text{obj[]} \mid T) \]

\[ \vdash \neg \Diamond \Box (\text{obj[]} \mid T) \]

Hence: \[ Im \mid Ir \vdash F \]

\[ A \land \neg A \vdash F \]
Example: Thief!

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

\[
\begin{align*}
\text{Shopper} & \triangleq \\
& Person[Wal\text{let}[$] \mid T] \land \\
& \Diamond(Person[No\text{Wallet}] \mid Wallet[$])
\end{align*}
\]

\[
\begin{align*}
\text{No\text{Wallet}} & \triangleq \neg(Wallet[$] \mid T)
\end{align*}
\]

\[
\begin{align*}
\text{Thief} & \triangleq Wallet[$] \triangleright \Diamond\text{No\text{Wallet}}
\end{align*}
\]

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

\[
\begin{align*}
\text{Shopper} \parallel \text{Thief} & \Rightarrow \\
& (Person[Wal\text{let}[$] \mid T] \mid \text{Thief}) \land \\
& \Diamond(Person[No\text{Wallet}] \mid No\text{Wallet})
\end{align*}
\]
**Fresh-Name Quantifier**

\[ P \models \forall x. A \iff \exists m \in \Lambda. m \notin fn(P, A) \land P \models A[x \leftarrow m] \]

- C.f.: \( P \models \exists x. A \iff \exists m \in \Lambda. P \models A[x \leftarrow m] \)
- Actually definable (metatheoretically, as an abbreviation):

\[ \forall x. A \iff \exists x. x \#(fn(A)-\{x\}) \land x \otimes T \land A \]

- **Fundamental “freshness” property (Gabbay-Pitts):**

\[ \forall x. A \iff \exists m \in \Lambda. m \notin fn(P, A) \land P \models A[x \leftarrow m] \]

\[ \iff \forall m \in \Lambda. m \notin fn(P, A) \Rightarrow P \models A[x \leftarrow m] \]

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

- **Very nice properties:**
  - \( \forall x. A \Rightarrow \forall x. A \Rightarrow \exists x. A \)
  - \( \neg \forall x. A \iff \forall x. \neg A \)
  - \( \forall x.(A \parallel B) \iff (\forall x. A) \parallel (\forall x. B) \)
  - \( \Diamond \forall x. A \iff \forall x. \Diamond A \)
*Hidden-Name Quantifier*

\[(\forall x)A \iff \forall x.x@A\]

- **Example:** \((\forall x)x[T] = \forall x.x@x[T]\)
  - “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 \(x\)”
  - \((\forall n)n[] \vdash (\forall x)x[T]\) because \((\forall n)n[] \vdash \forall x.x@x[T]\)
    - because \((\forall n)n[] \vdash n@n[T]\) (where \(n \notin \text{fn}(\forall n)n[]\)).

- **Other examples**
  - \((\forall m)m[] \vdash (\forall x)n[]\)
  - \((\forall n)n[] \vdash (\forall n)n[] \nmid (\forall x)(x[] \mid x[])\)
  - \((\forall n)(n[] \mid n[]) \nmid (\forall x)x[] \mid (\forall x)x[]\)
A Good Property

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

\[(\forall x)(\mathcal{A}\{n \leftarrow x\}) \land n \oplus T \vdash n \oplus \mathcal{A} \quad \text{where } x \notin \text{fv}(\mathcal{A})\]

It implies:

\[P \models \mathcal{A} \Rightarrow (\forall n)P \models (\forall x)(\mathcal{A}\{n \leftarrow x\})\]

\[P \models n \oplus \mathcal{A} \Rightarrow P \models (\forall x)(\mathcal{A}\{n \leftarrow x\})\]

\[P \models (\forall x)(\mathcal{A}\{n \leftarrow x\}) \land n \notin \text{fn}(P) \Rightarrow P \models n \oplus \mathcal{A}\]
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”.

$$(\forall x) \ (n[\otimes x] \parallel m[\otimes x])$$

• $P \vdash (\forall x) \ (n[\otimes x] \parallel m[\otimes x])$

$$\iff \exists r \in \Lambda. \ r \notin fn(P) \cup \{n,m\} \land \exists R', R'' \in \Pi. \ P \equiv (\forall r)(n[R'] \parallel m[R'']) \land r \in fn(R') \land r \in fn(R'')$$

• E.g.: take $P = (\forall p) \ (n[p][] \parallel m[p][])$.

• A protocol establishing a shared key should satisfy:

$$\Diamond (\forall x) \ (n[\otimes x] \parallel m[\otimes x])$$
Applications

- Verifying security+mobility protocols.
- Modelchecking security+mobility assertions:
  - If $P$ is $!$-free and $\mathcal{A}$ is $\triangleright$-free, then $P \models \mathcal{A}$ 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.